Polynomials in the Bernstein Basis and Their Use in Stability Analysis

Leth, Tobias

DOI (link to publication from Publisher):
10.5278/vbn.phd.tech.00024

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

POLYNOMIALS IN THE BERNSTEIN BASIS AND THEIR USE IN STABILITY ANALYSIS

BY
TOBIAS LETH

DISSERTATION SUBMITTED 2017

AALBORG UNIVERSITY
DENMARK

Polynomials in the Bernstein Basis and Their Use in Stability Analysis

PhD Dissertation
Tobias Leth

Dissertation submitted October 13, 2017

Dissertation submitted:	October 13, 2017
PhD supervisor:	Prof. Rafał Wisniewski Aalborg University
Assistant PhD supervisor:	Assoc. Prof. Christoffer Sloth Aalborg University
PhD committee:	Associate Professor Henrik Schiøler (chairman) Aalborg University
	Associate Professor Raphaël Jungers UCLouvain
	Software Engineer Joachim Dahl Mosek ApS
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-087-6

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2017

Abstract

This Thesis considers stability analysis of polynomial dynamical systems using formulations in the Bernstein basis. The thesis is split into two parts, one considering the simplicial Bernstein basis and one considering its utilisation in stability analysis.

The first three chapters cover the notation and definitions needed to introduce the Bernstein basis polynomials, how polynomials are described in the basis, conventions for numbering simplices and vertices, some properties of the basis polynomials, and the arithmetic needed in part two. The last chapter of part one considers a utilisation of the Bernstein basis to certify positivity of positive polynomials.

Part two is dedicated to stability analysis in the sense of Lyapunov. The first chapter considers polynomial Lyapunov functions described in the Bernstein basis and how the conditions on the functions translates into conditions on the coefficients when described in the basis. The remaining chapters develop algorithms for the automated generation of Lyapunov functions and focuses on convergence, solvability, and strategies for modification of unsolvable linear programming problems.

Resumé

I denne afhandling benyttes beskrivelser i Bernstein basen til at analysere stabilitet af dynamiske systemer med polynomiel karakteristik. Afhandlingen er i to dele, den ene handler om den simplicielle Bernstein base og den anden anvender basen til stabilitets analyse.

De første tre kapitler omhandler notation og definitioner nødvendige til indførelsen af Bernstein basis polynomierne, hvordan polynomier er beskrevet i basen, konventioner til nummerering af simplekser og vertekser, nogle egenskaber ved base polynomierne og arimetik der skal bruges i del to. Det sidste kapitel i del et behandler anvendelsen af Bernstein basen til at certificerer positive polynomier positive.

Del to betragter stabilites analyse i henhold til Lyapunov. Det første kapitel omhandler polynomielle Lyapunov funktioner beskrevet i Bernstein basen og hvordan betingelserne på funktionerne kan oversættes til betingelser på koefficienterne. De resterende kapitler udvikler algoritmer til automatisk genereing af Lyapunov funktioner og fokuserer på konvergens, løselighed og strategier til modifikationen af uløselige lineære programmerings problemer.

Acknowledgements

Several individuals and institutions deserve acknowledgement for their contributions to my studies and time as a PhD student.

First and foremost, my gratitude goes to my supervisors, Rafał Wisniewski and Christoffer Sloth. They initiated the process of funding the project long before I finished my master studies, and when time came, they accepted my application for the position. During the PhD, they continuously challenged my theoretic understanding and they aided and guided my work in whatever direction I found it interesting or necessary to proceed. I feel honoured to have been allowed to submerge into the depths that are scientific research.

During regular workdays, the atmosphere at my department was predominantly friendly and easy going. When I needed help, advise, or simply a break, I often sought out colleagues and I was rarely turned away. Especially my workout buddy Rasmus Pedersen and the members of my lunch club John-Josef Leth, Henrik Schiøler, Jesper Dejgaard Pedersen, and Simon Jensen, deserve to be mentioned by name. They provided an often much needed break from the studies. Also the section administrator, Susanne Nørrevang, has been invaluable. She knows the ins and outs of the bureaucracy and was always quick to provide whatever document or connection needed to resolve any and all problems I had during the past three years.

My appreciation goes to Jan Østergaard who spent hours pondering and discussing my questions regarding information hidden in unsolvable linear problems. Without his help, I am not sure I would have found Farkas' Lemma.

Outside Aalborg University, Sriram Sankaranarayanan deserves praise. He hosted me during my exchange visit at the University of Colorado at Boulder. We had numerous fruitful discussions expanding my understanding of both his and my own work. During my exchange visit, Lieven Vandenberghe invited me to the University of California at Los Angeles and Kathryn Johnson invited me to the National Renewable Energy Laboratory. I extend my gratitude to the both of them.

The last persons to thank are the fourth person involved in the CodeMe project, Marie-Françoise Roy and her husband Michel Coste. I thank them

for their valuable inputs to the more mathematical aspects of my Thesis.

Regarding institutions, I would like to thank the following three foundations for financial support at various occasions during my studies: Augustinus Fonden, Otto Mønsted, and Fabrikant P. A. Fiskers Fond.

Finally, I thank Klitgården Refugium for hosting me during the final phase of writing my Thesis.

Tobias Leth
Klitgården, September 18, 2017

Contents

Preface

This Thesis is submitted as a monograph in partial fulfilment for the degree of Doctor of Philosophy at the Section of Automation and Control, Department of Electronic Systems, Aalborg University, Denmark. The work has been conducted from October 15th, 2014 to October 13th, 2017 under the supervision of Professor Rafał Wisniewski and Associate Professor Christoffer Sloth. The work was supported by the Danish Council for Independent Research under grant number DFF - 4005-00452 in the project CodeMe. From September 6th, 2016 to January 14th, 2017 I lived in Boulder, Colorado, USA and studied under the supervision of Associate Professor Sriram Sankaranarayanan.

During my studies, I felt like I tried to answer a question that I did not know how to ask. As I dug deeper into the subject, the quantities of interest revealed themselves and the formulation of the right questions started to take place. The notation had to be changed a couple of times to accommodate this. The notation presented in the first few chapters is perhaps not intuitive at first glance, but as you read on, dear reader, I am comfortable it will make sense to you.

This Thesis is atypical in the sense that I will take the reader on a journey through the entirety of my studies; my results, my failures, and the as-of-yet unanswered questions. While studying, I did sometimes ponder quite a lot on the questions which did not allow for a clean mathematical answer. I allowed for much of this pondering to find its way into the thesis as I find it interesting and necessary to fully convey my considerations and explorations into the subjects.

Tobias Leth
Klitgården, September 18, 2017

Introduction

This Chapter is included to serve as short introduction into why the content of my PhD study is interesting. It also presents an overview of my contributions in the field of stability analysis, outreach to related work, and an outline of the content.

Motivation

The study presented in this Thesis is motivated by the simple goal of further automating the application of Lyapunov theory. Linear systems are well understood and their analysis is accompanied by software tools for stability investigation and controller derivation. The next natural step is to attack the non-linear case. This has shown itself to be significantly more involved than the linear counterpart.

Arguably, the softest non-linearity is the class of polynomials. This is why the vast majority, if not all, of the existing material on the subject is restricted to polynomial non-linearities. Most of the existing software tools rely on semi-definite programming which suffers from scalability issues. In addition, the existing software tools are predominantly academic in the sense that they require a user with significant insight and theoretical knowledge. Compared to the results obtained for linear systems, the work on polynomial systems has just begun and there is a difficult and long road ahead.

To take a step in that direction, the project CodeMe set out to utilise specific properties of the Bernstein basis. By understanding the basis and linking it to Lyapunov theory, a technique for stability analysis relying on linear programming could potentially be obtained. It was expected to be able to derive linear algorithms to aid the analysis and design of complex systems. It was also expected to devise a software tool applicable in the industry by users without expert knowledge on the subject.

It is left to the reader to pass judgement on whether or not these goals have been reached.

Contributions

My studies can be divided into three areas.

First, I tackled a case study on the use of positive and sum-of-squares polynomials to prove stability in a drinking water distribution network. This was useful to familiarise myself with state-of-the-art tools and techniques. The work was published in [30].

Secondly, I worked on understanding the Bernstein basis, particularly its advantages and drawbacks in relation to stability analysis. It initially resulted in [27] including an algorithm for synthesising Lyapunov functions. Afterwards, the results were improved and extended into [29] where the algorithm was modified to efficiently handle infeasibility.

Thirdly, I considered termination criteria for the algorithm. I was especially interested in the circumstances under which termination could be compromised even when the algorithm was applied to a stable system. This called for the resurrection of the notion of rational stability and resulted in [28].

Throughout my studies, I implemented functions to execute various calculations. The functions have been composed into a MATLAB toolbox accompanied by a tutorial and it is freely available. It can be downloaded from the project web page: http://kom.aau.dk/project/CodeMe/software.html

Related Work

Being a monograph, the thesis does not contain an explicit state-of-the-art section. Readers interested in related work are referred to the following sections and external references on various topics.

This Thesis solely focuses on the simplicial Bernstein basis. Readers interested in the tensorial description are referred to e.g. [8] or [15]. Readers interested in the description of general convex polytopes using the Handelman basis are referred to [21].

The Bernstein basis has found its use in other applications. See e.g. [35] or [36] for applications in optimisation, [50] for applications in reachability, or [51] for applications in polynomial evaluation.

The introduction in Chapter Positivity Certification contains references to other techniques for positivity certification.

Section 17.3 contains references to work regarding weakening the requirements on Lyapunov functions and using the Dini derivative.

Section 18.3 contains a collection of results regarding existence of structured Lyapunov functions.

Section 22 presents an alternative utilisation of the Bernstein basis for synthesising Lyapunov functions presented in [42]. The perhaps most widely known approach for synthesising Lyapunov functions is the sum-of-squares realisation. It first appeared in [37] and is accompanied by the toolbox SOSTOOLS [40]. The toolbox GloptiPoly [19] is another option which uses the moment approach to formulate the optimisation.

Outline

The thesis is divided into two parts. The first part is concerned with the description of polynomials in the simplicial Bernstein basis. It collects information from throughout the literature and provides a uniform and coherent presentation of known and new results. Especially the extension to working on more than one simplex is novel, and needed in part two. Part two considers the use of the Bernstein basis in stability analysis using Lyapunov functions. It derives theory and algorithms for the synthesis of Lyapunov functions and particular emphasis is placed on handling infeasibility efficiently.

Chapters presenting mathematical operations terminate with a section covering software, implemented in MATLAB, to perform the operations. A tutorial to the toolbox is given in the last appendix.

Part I

Polynomials in the Bernstein Basis

Bernstein Basis Introduction

This Chapter introduces all basic concepts needed to present the Bernstein basis polynomials. Afterwards, polynomials are described on one simplex and what is later introduced as a collection of simplices.

1 Definitions

Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ denote the natural, integer, rational, and real numbers, $\mathbb{N}_0$ denotes $\mathbb{N} \cup \{0\}$, and $\mathbb{R}_{>0}$ ($\mathbb{R}_{\geq 0}$) denotes the positive (non-negative) real numbers.

$\mathbb{R}[x]$ denotes the ring of polynomials with real coefficients, $\mathbb{R}_{>0}[x]$ and $\mathbb{R}_{\geq 0}[x]$ are the positive (semi-)definite polynomials, and $\Sigma[x]$ are the polynomials which can be represented as a sum of squares. Polynomials will be in n variables, with n obvious from context.

Let $p \in \mathbb{R}[x]$. Then both $p(\hat{x})$ and $p|_{\hat{x}}$ denote the evaluation of p at the point $\hat{x}$.

Define the interior of a ball in $\mathbb{R}^n$ of radius r by $B_r = \{x \in \mathbb{R}^n : ||x||_2 < r\}$, where $\mathbb{R}^n$ is the standard n-dimensional Euclidean space. The closure of the ball is written $\bar{B}_r = \{x \in \mathbb{R}^n : ||x||_2 \leq r\}$. The ball of radius $r = 1$, termed the unit ball, is written simply as B ($\bar{B}$).

Define the unit cube in $\mathbb{R}^n$ by $B^\infty = \{x \in \mathbb{R}^n : ||x||_\infty \leq 1\}$.

Define unit vectors e_j as vectors of length $n+1$ with zeros on all entries except the $(j+1)^{\text{th}}$ like

$$e_j = [\underbrace{0, \ldots, 0}_{j \text{ 0s}}, 1, 0, \ldots, 0] \qquad \forall\, j \in \{0, 1, \ldots, n\}. \tag{1}$$

If A is an $n \times m$ matrix, A^T denotes its $m \times n$ transpose. Vectors will usually be column vectors, but sometimes explicitly expressed as row vectors. This is to avoid excessive use of the transpose operator.

1.1 Barycentric Coordinates

The Bernstein basis polynomials used in this thesis are defined on simplices, which in turn are described by their barycentric coordinates.

Let $\sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{R}^n$ be $n+1$ affinely independent points and let $\lambda_0, \lambda_1, \ldots, \lambda_n \in \mathbb{R}[x]$ be affine polynomials. Then, if

$$x = \lambda_0(x)\sigma_0 + \lambda_1(x)\sigma_1 + \cdots + \lambda_n(x)\sigma_n \qquad \forall\, x \in \mathbb{R}^n \tag{2}$$

$\lambda_i(x)$ are the barycentric coordinates associated to σ_i. Henceforth I denote $\lambda_i = \lambda_i(x)$ and omit the dependency on the variables. Note that barycentric coordinates are not unique. If, in addition

$$\sum_{i=0}^{n} \lambda_i = 1, \tag{3}$$

λ_i are unique, and are sometimes referred to as absolute barycentric coordinates. In the rest of the thesis, only the unique version is used, and it will be referred to as barycentric coordinates. Combining the above yields

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \sigma_1(1) & \cdots & \sigma_n(1) \\ \sigma_0(2) & \sigma_1(2) & \cdots & \sigma_n(2) \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_0(n) & \sigma_1(n) & \cdots & \sigma_n(n) \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \cdots \\ \lambda_n \end{bmatrix}. \tag{4}$$

Example

Consider the points $\sigma_0 = (0,0)$, $\sigma_1 = (4,0)$, and $\sigma_2 = (0,2)$. If a point P_1 in $\mathbb{R}^2$ has Cartesian coordinates $P_1(x_1, x_2) = (1,1)$, it has barycentric coordinates $P_1(\lambda_0, \lambda_1, \lambda_2) = (0.25, 0.25, 0.5)$ associated to σ_i.

Similarly, the points $P_2(x_1, x_2) = (0, 0.5)$ and $P_3(x_1, x_2) = (3,2)$ have barycentric coordinates $P_2(\lambda_0, \lambda_1, \lambda_2) = (0.75, 0, 0.25)$ and $P_3(\lambda_0, \lambda_1, \lambda_2) = (-0.75, 0.75, 1)$ associated to σ_i. The points are shown in Figure 1.

■

1.2 Simplex & Collection of Simplices

With the definition of barycentric coordinates in place, I can now introduce the notion of a simplex.

Let $\sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{R}^n$ be $n+1$ affinely independent vertices with ordering $<$ such that $\sigma_i < \sigma_j$ if $i < j$. A non-degenerate n-simplex (a simplex of dimension n) $\sigma \equiv [\sigma_0, \sigma_1, \ldots, \sigma_n]$ is the convex hull created by $n+1$ affinely

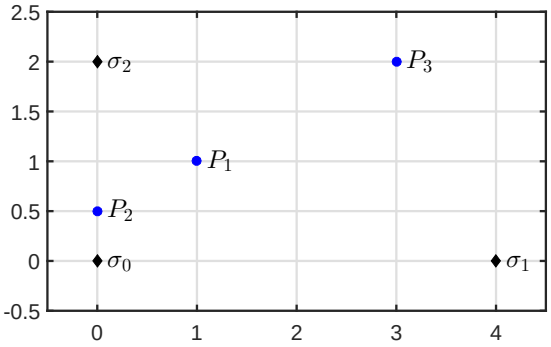


Figure 1: Example of barycentric coordinates for the points P_1, P_2, and P_3 associated with σ_0, σ_1, and σ_2. Note that the σ_i's are affinely independent whereas the P_i's are not.

independent vertices σ_i such that

$$\sigma = \left\{ \sum_{i=0}^{n} \lambda_i \sigma_i | \lambda_i \geq 0, i = 1, \ldots, n \right\} \subset \mathbb{R}^n, \tag{5}$$

where the λ_i's are the barycentric coordinates for σ. Comparing to barycentric coordinates, the σ_i's are now not just points, but vertices, which can be thought of as corners of something. This is intuitive, since a simplex is the convex hull of its defining vertices, which is also seen in Equation (5) from the $\lambda_i \geq 0$ requirement. Again, comparing to barycentric coordinates and the above example, it is seen that the point P_2 is located on the boundary of the simplex defined by σ_0, σ_1, and σ_2 since one of its coordinates is zero. The point P_3 has a negative barycentric coordinate showing that the point does not belong to the simplex.

A simplex is degenerate if it is defined by $n+1$ vertices but is not of dimension n. This happens when the vertices are not affinely independent and thus span a lower dimensional space. In the rest of the thesis only non-degenerate simplices are used, and they will be referred to as simplices.

An l-face of a n-simplex $\sigma^1 = [\sigma_0^1, \sigma_1^1, \ldots, \sigma_n^1]$, with $l \leq n$, is any l-simplex $\sigma^2 = [\sigma_0^2, \sigma_1^2, \ldots, \sigma_l^2]$ with ordering $<$, such that

$$\{\sigma_0^2, \sigma_1^2, \ldots, \sigma_l^2\} \subseteq \{\sigma_0^1, \sigma_1^1, \ldots, \sigma_n^1\}. \tag{6}$$

The n-face of an n-simplex is the simplex itself, $(n-1)$-faces of an n-simplex are called facets and 0-faces are simply the vertices. Some authors call 1-faces edges, but the term edge is not used in this Thesis.

Frequently, I will need to refer to different facets and faces of simplices. Defining a face operator as

$$\partial_i^{n+1} \circ \sigma = [\sigma_0, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n] \tag{7}$$

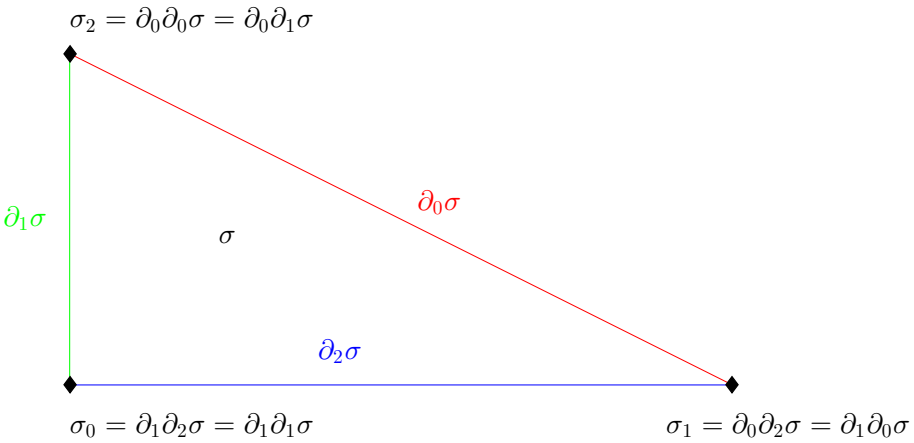


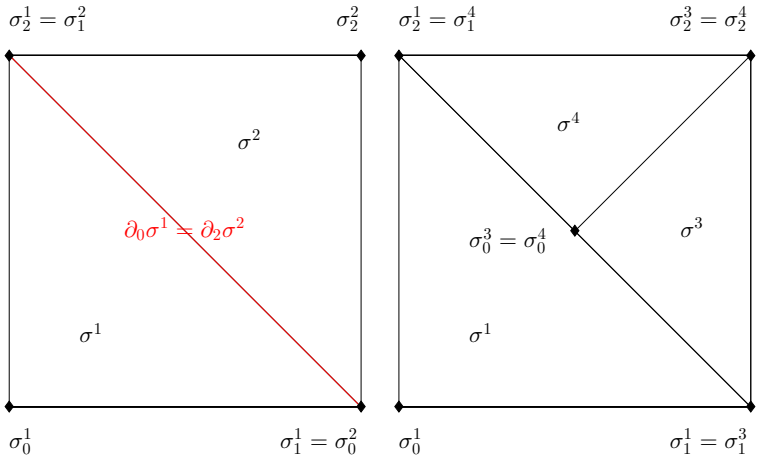
Figure 2: Simplex σ defined by the vertices σ_0, σ_1, and σ_2. The face operator ∂ can be used to identify the faces of σ.

and writing $\partial_i^{n+1}\sigma = \partial_i^{n+1} \circ \sigma$ the facets of σ can be identified. Consecutive application of ∂ will identify the lower dimensional faces and, if applied n times, the vertices. In the rest of the thesis, the superscript of ∂_i^{n+1} is dropped and the dimension, of the simplex on which ∂ operates, will be obvious from context. Figure 2 shows the vertices, the simplex, and how the face operator can be used to identify the faces. Note that the numbering of the vertices is arbitrary. Also note the ambiguity in the way the low dimension faces can be obtained through different sequencing of applying ∂_i. This effect becomes increasingly more involved when n grows, but care will be taken to ensure unambiguity when needed.

Later in the thesis, I will often need to work on more than one simplex. The familiar reader will recognise the object called a simplicial complex as suited for this purpose, see [7] for details, but actually a simplicial complex contains too much information since I do not need the simplices of dimension lower than n. For this reason I define the following.

Definition 1 *(Collection of Simplices) Let $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ be a finite set of m non-overlapping (except for faces) n-simplices, and let them be ordered by $<$ such that $\sigma^i < \sigma^j$ if $i < j$. Then K is called a collection of simplices.*

Figure 3 shows two collections of simplices. The non-overlapping part in Definition 1 can be seen on Figure 3(a) where $K = \{\sigma^1, \sigma^2\}$ and σ^1 and σ^2 share the facet $\partial_0\sigma^1 = \partial_2\sigma^2$, and nothing more. On Figure 3(b) σ^2 has been split in into two simplices resulting in the collection $K = \{\sigma^1, \sigma^3, \sigma^4\}$. This results in a non-proper collection of simplices. A collection of simplices K is termed non-proper if any l-face of σ^i overlaps with any k-face of σ^j where $\{\sigma^i, \sigma^j\} \subseteq K$ and $l \neq k$. This is the case on Figure 3(b) for $\partial_0\sigma^1$ and σ_0^3.



(a) Proper collection of simplices. (b) Non-proper collection of simplices.

Figure 3: Proper and non-proper collections of simplices. Non-overlapping except on the faces.

Only rarely will I utilise non-proper collections of simplices. Thus, in the rest of the thesis the property of being proper is assumed for all collections of simplices and not mentioned explicitly.

For completeness, Figure 4 shows two instances of simplices which cannot constitute a collection of simplices. In Figure 4(a) the two simplices are overlapping with more than just faces. In Figure 4(b) the simplices are not connected. This situation is uninteresting when it comes to stability analysis and it will never occur in this thesis.

Occasionally, the size of a simplex or the size of simplices in a collection of simplices is of interest. This size is captured by the diameter of a simplex.

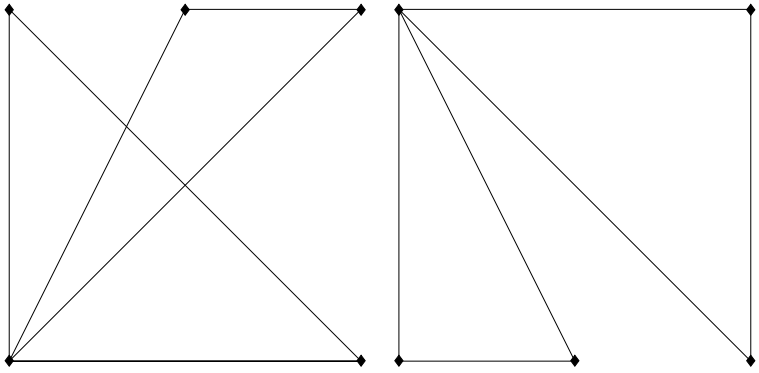
Definition 2 *Let the diameter of a simplex σ be the maximal distance between any two points in the simplex, and denote the diameter of σ by $h(\sigma)$. For the collection $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ define*

$$h(K) = \max_{\sigma \in K} h(\sigma) \tag{8}$$

to be the maximal diameter of any simplex in K.

1.3 Triangulation

Later on I will need to define polynomials on box polytopes. The mathematics revolving triangulation of polytopes is rich and involved, but for my purpose I only need the most simple and intuitive concepts. The interested reader is referred to [7], but a triangulation of a polytope can be thought



(a) Overlapping simplices. (b) Disconected simplices.

Figure 4: Two cases of simplices which cannot constitude a collecection of simplices.

of as chopping the polytope into simplices, which then together constitute a collection of simplices. To this end, I use the so-called Kuhn's triangulation.

Definition 3 *(Kuhn's Triangulation [26]) Let Θ^n be the set of all permutations of n elements, and note that there are $m = n!$ permutations in Θ^n. For any permutation $\theta \in \Theta^n$ define the simplex*

$$\sigma^\theta = \{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n | -1 \leq x_{\theta(n)} \leq x_{\theta(n-1)} \leq \cdots \leq x_{\theta(1)} \leq 1\}. \quad (9)$$

Then the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ is a triangulation of the n-dimensional unit cube B^∞ called Kuhn's Triangulation.

For reasons which will become evident later in Chapter Bernstein Basis Properties and Chapter Stability Introduction am I interested in the case where the origin is in the interior of the polytope, and I need a vertex to be placed at the origin. To obtain this, Kuhn's triangulation can be used on the $2n$ facets of B^∞ resulting in $2n!$ $(n-1)$-simplices. Adding the origin to all of the simplices results in $2n!$ n-simplices. In this case I say that the collection K covers B^∞. Figure 5 shows this in the 3 dimensional case where six of the resulting simplices are omitted and the remaining six simplices are exploded for clarification.

1.4 Basis Polynomials

Having thoroughly introduced the notion of a simplex and collection of simplices, and how to obtain them, it is time to present the core element in this thesis; the Bernstein basis. I assume the reader to be familiar with the monomial basis for describing polynomials. I will, however, briefly refresh the

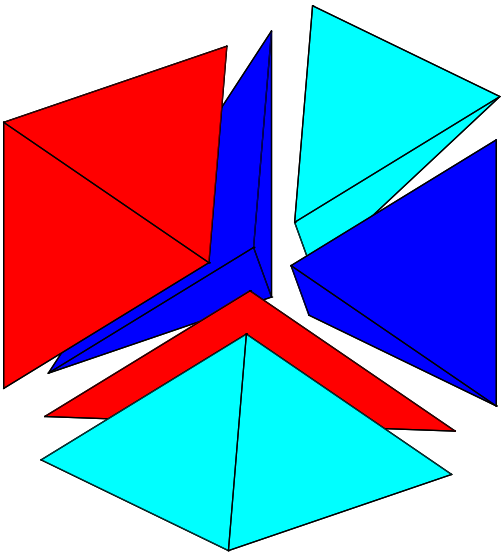


Figure 5: Six of the 12 simplices obtained from applying Kuhn's triangulation on the facets of a 3 dimensional cube and adding the origin to each of them. The view is exploded for clarity.

monomial basis to make the transition to the Bernstein basis more easy and to present a convention on the representation of polynomials in the monomial basis for later transformation between the bases.

For $x \in \mathbb{R}$ the monomials $\{x^0, x^1, x^2\}$ constitute a basis for describing polynomials in one variable of degree two or less. A polynomial of degree two or less is then a linear combination of the elements in the basis. E.g. the coefficients $[c_0, c_1, c_2] = [-1, 0, 6]$ defines the polynomial $p(x) = 6x^2 - 1$. Writing $c = [-1, 6]$ and $\gamma = [0, 2]$, p can be defined as

$$p(x) = \sum_{i=1}^{2} c_i x^{\gamma_i}. \tag{10}$$

In general, for $x \in \mathbb{R}^n$ polynomials in the monomial basis will be defined as

$$p(x) = \sum_{k=1}^{K} c_k x^{\gamma_k} \tag{11}$$

where $c = [c_1, c_2, \ldots, c_K]$ is the coefficient vector and

$$\gamma = \begin{bmatrix} \gamma_1(1) & \gamma_2(1) & \cdots & \gamma_K(1) \\ \gamma_1(2) & \gamma_2(2) & \cdots & \gamma_K(2) \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_1(n) & \gamma_2(n) & \cdots & \gamma_K(n) \end{bmatrix} \tag{12}$$

$$x^{\gamma_k} = x_1^{\gamma_k(1)} x_2^{\gamma_k(2)} \cdots x_n^{\gamma_k(n)}. \tag{13}$$

The monomial basis is the most intuitive basis for polynomials since the elements of the basis are as simple as they can be. The elements in the Bernstein basis are significantly more involved, but the Bernstein basis offers properties which the monomial basis does not possess. These properties are covered in the next chapter, and they are the main argument behind working in the Bernstein basis.

Define a multi-index $\alpha \in \mathbb{N}_0^{n+1}$ and let

$$|\alpha| = \sum_{i=0}^{n} \alpha_i, \qquad \lambda^\alpha = \prod_{i=0}^{n} \lambda_i^{\alpha_i}, \qquad \binom{D}{\alpha} = \frac{D!}{\alpha_0!\alpha_1! \cdots \alpha_n!}, \tag{14}$$

where λ_i are the barycentric coordinates to simplex σ.

Definition 4 *(Bernstein Basis Polynomials [26]) The Bernstein basis polynomials of degree D on simplex σ are defined as*

$$\mathcal{B}_\alpha^D(\sigma) = \binom{D}{\alpha} \lambda^\alpha, \qquad \forall |\alpha| = D, \tag{15}$$

and $\mathcal{B}^D(\sigma)$ is a column vector containing all the Bernstein basis polynomials of degree D on σ.

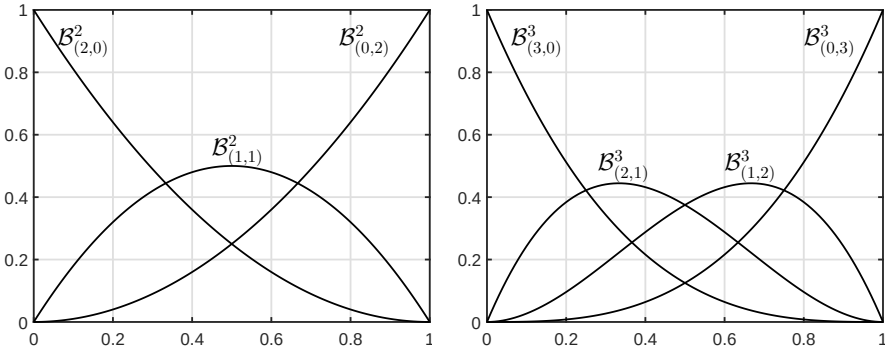
Note that the barycentric coordinates λ are defined by the vertices of σ, which in turn define σ. This explains the choice of writing the basis polynomials as dependent on σ. I will alternate between $\mathcal{B}^D$ and $\mathcal{B}^D(\sigma)$ throughout the thesis whenever simplicity of notation does not interfere with clarity.

Later, the order of appearance of the possible multi-index combinations in $|\alpha| = D$ will become important. For this reason, I impose the Lexicographic order on the numbering of the vertices according to $>$ which then also applies to the different α combinations. As an example consider $n = 2$ and $D = 2$ for which the possible combinations are

$$\begin{array}{c} \\ \sigma_0 \\ \sigma_1 \\ \sigma_2 \end{array} \begin{array}{c} \begin{array}{cccccc} \alpha = (2,0,0) & \alpha = (1,1,0) & \alpha = (1,0,1) & \alpha = (0,2,0) & \alpha = (0,1,1) & \alpha = (0,0,2) \end{array} \\ \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 \end{bmatrix} \end{array}, \tag{16}$$

where, as a result of the Lexicographic order and $>$ on the vertices, $\alpha = (2,0,0) > \alpha = (1,1,0) > \alpha = (1,0,1)$ and so on. The matrix in Equation (16) will be referred to as the α-matrix. Denote the number of possible combinations as

$$N_D = \binom{n+D}{n}. \tag{17}$$



(a) One dimensional Bernstein basis polynomials of degree 2.

(b) One dimensional Bernstein basis polynomials of degree 3.

Figure 6: One dimensional Bernstein basis polynomials on the simplex $\sigma = [0, 1]$.

The choice of having the barycentric coordinates sum to one carries over to the basis polynomials which also sum to one. This can be seen by expanding using the multinomial theorem as

$$1 = 1^D = \left(\sum_{i=0}^{n} \lambda_i\right)^D = \sum_{|\alpha|=D} \binom{D}{\alpha} \prod_{i=0}^{n} \lambda_i^{\alpha_i} = \sum_{|\alpha|=D} \mathcal{B}^D_\alpha. \tag{18}$$

Example

In one dimension, on the simplex $\sigma = [0, 1]$ the barycentric coordinates are $\lambda_0 = (1 - x)$ and $\lambda_1 = x$. The Bernstein basis polynomials of degree 2 are

$$\mathcal{B}^2_{(2,0)} = (1-x)^2, \qquad \mathcal{B}^2_{(1,1)} = 2x(1-x), \qquad \mathcal{B}^2_{(0,2)} = x^2. \tag{19}$$

On the same simplex, the Bernstein basis polynomials of degree 3 are

$$\mathcal{B}^3_{(3,0)} = (1-x)^3, \qquad \mathcal{B}^3_{(2,1)} = 3x(1-x)^2, \tag{20}$$

$$\mathcal{B}^3_{(1,2)} = 3x^2(1-x), \qquad \mathcal{B}^3_{(0,3)} = x^3. \tag{21}$$

Figure 6 shows the basis polynomials plotted on their defining simplex.

Let $x = (x_1, x_2) \in \mathbb{R}^2$. In 2 dimensions, on the simplex $\sigma = [\sigma_0, \sigma_1, \sigma_2]$ with $\sigma_0 = [0, 0]^T$, $\sigma_1 = [1, 0]^T$, and $\sigma_2 = [0, 1]^T$, the barycentric coordinates are $\lambda_0 = (1 - x_1 - x_2)$, $\lambda_1 = x_1$, and $\lambda_2 = x_2$. The Bernstein basis polynomials of degree 2 are

$$\mathcal{B}^2_{(2,0,0)} = (1 - x_1 - x_2)^2, \qquad \mathcal{B}^2_{(1,1,0)} = 2x_1(1 - x_1 - x_2), \tag{22}$$

$$\mathcal{B}^2_{(1,0,1)} = 2x_2(1 - x_1 - x_2), \qquad \mathcal{B}^2_{(0,2,0)} = x_1^2, \tag{23}$$

$$\mathcal{B}^2_{(0,1,1)} = 2x_1x_2, \qquad \mathcal{B}^2_{(0,0,2)} = x_2^2. \tag{24}$$

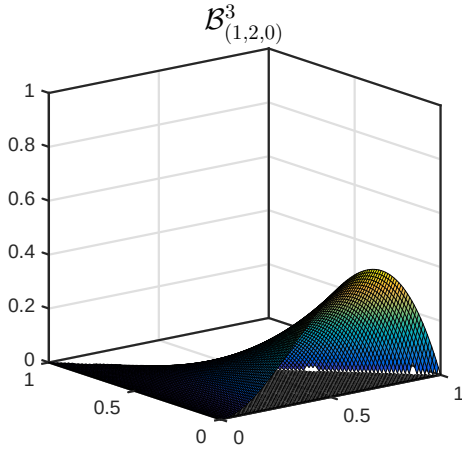


Figure 7: The Bernstein basis polynomial $\mathcal{B}^3_{(1,2,0)}$. See Figure A.2 for the remaining 9 basis polynomials for $n = 2$, $D = 3$. The shaded area is the simplex.

In a given dimension n, for a given degree D there are N_D basis polynomials. In Appendix A the basis polynomials for $n = 2$, $D = 2$ and $n = 2$, $D = 3$ are plotted. Figure 7 shows one of the Bernstein basis polynomial in two dimensions of degree 3.

■

In the example, both $\sigma = [0, 1]$ in one dimension and $\sigma = [\sigma_0, \sigma_1, \sigma_2]$ with $\sigma_0 = [0, 0]^T$, $\sigma_1 = [1, 0]^T$, and $\sigma_2 = [0, 1]^T$ in two dimensions are the one dimensional and two dimensional standard simplex, respectively. In general, the standard simplex is given as

$$[0_{n\times 1}, I_{n\times n}]\,. \tag{25}$$

2 Polynomials in the Bernstein Basis

Completely analogous to the monomial basis, any polynomial p of degree d in n variables can be described in the Bernstein basis on simplex σ of degree $D \geq d$ as a linear combination of the elements in the basis as

$$p(x) = \sum_{|\alpha|=D} b_\alpha(p, D, \sigma)\mathcal{B}^D_\alpha = b(p, D, \sigma)\mathcal{B}^D. \tag{26}$$

The vector $b(p, D, \sigma)$ is a row vector and it contains the Bernstein coefficients of p described in degree D on simplex σ, and it uniquely describes $p(x)$. The dependencies on p, D, and σ in $b_\alpha(p, D, \sigma)$ will be dropped for notational simplicity whenever clarity allows.

For each coefficient $b_\alpha(p, D, \sigma)$ a corresponding grid point is defined as

$$\Delta_\alpha(D, \sigma) = \frac{\alpha_0\sigma_0 + \alpha_1\sigma_1 + \cdots + \alpha_n\sigma_n}{D}. \tag{27}$$

The grid points are equally distributed linear combinations of the vertices normalised by the degree. Thus every coefficient is located somewhere in the simplex. Note that the grid point is not dependent on p. This is because the grid points are defined solely by which simplex they belong to, and the degree of the basis polynomials. The pair $(\Delta_\alpha(D, \sigma), b_\alpha(p, D, \sigma))$ is the control point associated to α.

The transformation from monomial basis to the Bernstein basis is a linear transformation and can be characterised by a matrix A defined from

$$\mathcal{B}^D = A^T\mathcal{X}, \tag{28}$$

where $\mathcal{X} = \{x_1^D, x_1^{D-1}x_2, \ldots, x_n, 1\}$ is the monomial basis of degree D. The entries in A can be found by expanding the equation, collecting terms of $\mathcal{X}$, and setting coefficients equal to zero. In general, this gives N_D^2 unknowns in N_D^2 equations grouped together in N_D equations each in N_D variables, where $N_D = \binom{n+D}{n}$. Having determined A, the Bernstein basis coefficients are then given as

$$b^T(p, D, \sigma) = Ac^T, \tag{29}$$

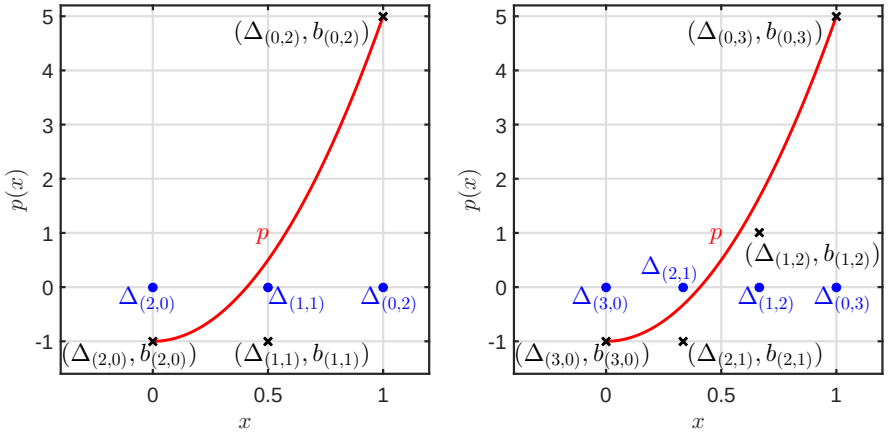
where c is the coefficient vector of p in the monomial basis. However, determining A in this manner can be tedious and another method for transforming from a description in the monomial basis to a description in the Bernstein basis on a given simplex is derived in Appendix B. It requires arithmetic for multiplying polynomials in the Bernstein basis and the so-called degree elevation. These operations are covered in Chapter Arithmetic.

The transformation between a description in the Bernstein basis on a given simplex to a description in the monomial basis is straight forward by simply expanding the expression and collecting terms of the elements in the monomial basis. This is show in the following example.

Example

The polynomial $p(x) = 6x^2 - 1$ from above is described in the Bernstein basis of degree 2 on the simplex $\sigma = [0, 1]$ by $b(p, 2, [0, 1]) = [-1, -1, 5]$. Expanding the description in the Bernstein basis reveals the description in the monomial basis as

$$p(x) = \sum_{|\alpha|=2} b_\alpha(p, 2, [0, 1])\mathcal{B}_\alpha^2 = -(1-x)^2 - 2x(1-x) + 5x^2 = 6x^2 - 1. \tag{30}$$



(a) The polynomial p with the grid points Δ_α and the coefficients in $b(p,2,\sigma)$.

(b) The polynomial p with the grid points Δ_α and the coefficients in $b(p,3,\sigma)$.

Figure 8: The polynomial $p(x) = 6x^2 - 1$ described in the Bernstein basis of degree 2 and 3 on the simplex $\sigma = [0,1]$.

If instead $p(x) = 6x^2 - 1$ is described in the Bernstein basis of degree 3 on the same simplex, the coefficient vector is $b(p,3,[0,1]) = [-1,-1,1,5]$. Again, by expanding

$$\begin{aligned} p(x) &= \sum_{|\alpha|=3} b_\alpha(p,3,[0,1])\mathcal{B}^3_\alpha \\ &= -(1-x)^3 - 3x(1-x)^2 + 3x^2(1-x) + 5x^3 = 6x^2 - 1. \end{aligned} \tag{31}$$

This expansion reveals an inherent feature of the Bernstein basis. The actual degree of a polynomial is not directly evident by looking at the coefficients. This is in contrast to the monomial basis where the degree is directly evident from the highest power of monomials with a non-zero coefficient.

Figure 8 shows p, the grid points, and the control points for the descriptions in degrees 2 and 3.

■

2.1 Basis Transformation and Additional Notation

A polynomial p defined in the Bernstein basis $\mathcal{B}^D(\sigma^i)$ on one simplex σ^i could be defined equally well on another simplex σ^j using $\mathcal{B}^D(\sigma^j)$. Generally, this makes $b(p,D,\sigma^i) \neq b(p,D,\sigma^j)$ and the change in basis is linear, thus

$$b^T(p,D,\sigma^i) = {}^iB_j b^T(p,D,\sigma^j) \tag{32}$$

where ${}^iB_j = I$ if $i = j$. Since the numbering of the vertices in a simplex is arbitrary, the order of the entries in b are not fixed until a choice is made on the numbering. Thus ${}^iB_j$ cannot be determined without knowing the numbering of the vertices in both σ^i and σ^j.

The matrix ${}^iB_j$ is characterised similarly to A from the transformation from monomial basis to the Bernstein basis. The entries of ${}^iB_j$ can be determined from

$$\mathcal{B}^D(\sigma^i) = {}^iB_j^T \mathcal{B}^D(\sigma^j) \tag{33}$$

by expanding the equation, collecting terms of $\mathcal{X}$, and setting coefficients equal to zero. Again this results in N_D^2 equations in N_D^2 unknowns. As before, determining ${}^iB_j$ in this manner can be tedious and another method for transforming between descriptions in the Bernstein basis on different simplices is derived in Appendix B.

Occasionally, it is convenient to identify the coefficients in $b(p, D, \sigma)$ which are restricted to a given face or often facet. By abuse of notation, I will reuse the face operator ∂ such that

$$\partial_k b(p, D, \sigma) = \{b_\alpha(p, D, \sigma) | \alpha(k+1) = 0\}, \tag{34}$$

for coefficients located on the facet $\partial_k \sigma$ and

$$\partial_{k_1}\partial_{k_2} b(p, D, \sigma) = \{b_\alpha(p, D, \sigma) | \alpha(k_1+1) = 0, \alpha(k_2+1) = 0\}, \tag{35}$$

for coefficients located on the $(n-2)$-face $\partial_{k_1}\partial_{k_2}\sigma$, and so on.

Example

Consider Figure 9, where two simplices are shown with vertex numbering and the grid points for polynomials of degree 3. In this case the α-matrix is

$$\begin{bmatrix} 3 & 2 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 2 & 3 \end{bmatrix}, \tag{36}$$

with the lexicographic order on the vertices imposed by $>$.

Since the simplices overlap on the facets $\partial_0\sigma^1$ and $\partial_2\sigma^2$ the grid points on $\partial_0\sigma^1$ and $\partial_2\sigma^2$ are defined by the same point in space, but the chosen numbering of the vertices makes the grid points coincide for different α-subscripts in the following pattern

$$\Delta_{(0,0,3)}(\sigma^1) = \Delta_{(0,3,0)}(\sigma^2), \qquad \Delta_{(0,1,2)}(\sigma^1) = \Delta_{(1,2,0)}(\sigma^2), \tag{37}$$

$$\Delta_{(0,2,1)}(\sigma^1) = \Delta_{(2,1,0)}(\sigma^2), \qquad \Delta_{(0,3,0)}(\sigma^1) = \Delta_{(3,0,0)}(\sigma^2). \tag{38}$$

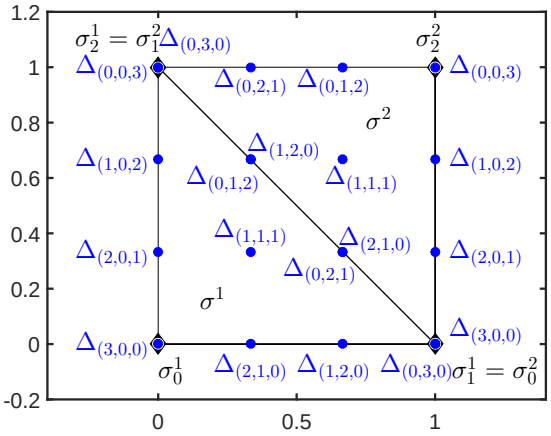


Figure 9: Collection of simplices $K = \{\sigma^1, \sigma^2\}$ with grid points $\Delta_\alpha(\sigma^i)$ for polynomials of degree 3. Note that the vertices are marked with a diamond coloured black and the grid points are marked with a circle coloured blue.

Now, consider the polynomial $p(x_1, x_2) = 6x_1^2 + 6x_2^2 - 1$ in the Bernstein basis of degree 3 on the simplex $\sigma^1 = [\sigma_0^1, \sigma_1^1, \sigma_2^1]$ with $\sigma_0^1 = [0,0]^T$, $\sigma_1^1 = [1,0]^T$, and $\sigma_2^1 = [0,1]^T$. It is described by the coefficient vector

$$b(p, 3, \sigma^1) = [-1, -1, -1, 1, -1, 1, 5, 1, 1, 5]. \tag{39}$$

Similarly, if defined on the simplex $\sigma^2 = [\sigma_0^2, \sigma_1^2, \sigma_2^2]$ with $\sigma_0^2 = [1,0]^T$, $\sigma_1^2 = [0,1]^T$, and $\sigma_2^2 = [1,1]^T$, p is described by the coefficient vector

$$b(p, 3, \sigma^2) = [5, 1, 5, 1, 3, 7, 5, 5, 7, 11], \tag{40}$$

with the transformation matrix from σ^1 to σ^2 as

$${}^2B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -2 & -2 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & -2 & -2 & 0 & 1 & 2 & 1 \\ -1 & 3 & 3 & -3 & -6 & -3 & 1 & 3 & 3 & 1 \end{bmatrix}. \tag{41}$$

The coefficients on the shared facet are identified as

$$\partial_0 b(\sigma^1) = [b_{(0,3,0)}(\sigma^1), b_{(0,2,1)}(\sigma^1), b_{(0,1,2)}(\sigma^1), b_{(0,0,3)}(\sigma^1)] = [5, 1, 1, 5] \tag{42}$$

and

$$\partial_2 b(\sigma^2) = [b_{(3,0,0)}(\sigma^2), b_{(2,1,0)}(\sigma^2), b_{(1,2,0)}(\sigma^2), b_{(0,3,0)}(\sigma^2)] = [5,1,1,5], \quad (43)$$

where the dependency on p and $D = 3$ has been omitted.

Note that

$$\partial_2 b(\sigma^1) = [b_{(3,0,0)}(\sigma^1), b_{(2,1,0)}(\sigma^1), b_{(1,2,0)}(\sigma^1), b_{(0,3,0)}(\sigma^1)] = [-1,-1,1,5], \quad (44)$$

which is the same coefficient vector which defined $p(x) = 6x^2 - 1$ in the one dimensional example from above. This is due to the fact that a polynomial restricted to one of its faces is defined by the coefficients on that face.

■

3 Polynomials on Collection of Simplices

With the description of polynomials on a single simplex in place, I can now introduce polynomials defined on a collection of simplices. However, as indicated by the example above, there are a number of redundant grid points in a collection of simplices, simply because every simplex gives rise to grid points which may already be determined from other simplices. This results in a lot of flexibility in terms of numbering and ordering. To streamline the process of working with polynomials defined in the Bernstein basis on collections of simplices, I introduce a few conventions for numbering and ordering (in addition to the $>$ on vertices and simplices from above).

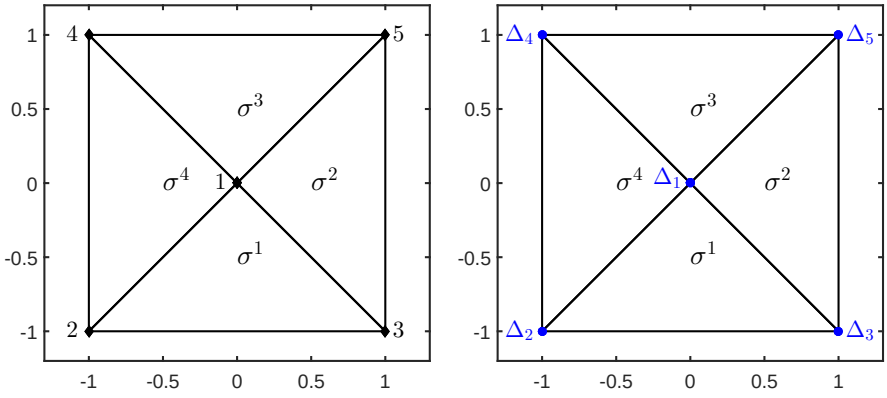
In the following $n = 2$ and $D = 2$ is used to exemplify the process, but the conventions are stated general.

Consider the 2-dimensional unit cube B^∞ covered by the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$ shown in Figure 10(a). With m the number of simplices in a collection, the numbering of the simplices are such that $\forall\ i \in \{1, \ldots, m-1\}$ simplices σ^i and σ^{i+1} always overlap with one facet. Which simplex to label as number 1 is still arbitrary.

In Figure 10(a) the 5 vertices defining the 4 simplices are labelled 1 to 5 such that the vertex at the origin is number 1. The remaining vertices are labelled arbitrarily. Labelling the origin as vertex 1 results in the following relation on the first grid point of all simplices which have a vertex at the origin

$$\Delta_{(D,0,0)}(\sigma^i) = \Delta_{(D,0,0)}(\sigma^j)\ \forall\ i, j \in \mathcal{I}, \quad (45)$$

where $\mathcal{I}$ is an index set of all simplices with the origin as a vertex. This relation is highly relevant for later algorithmic implementation.



(a) Unit cube covered by collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$.

(b) The numbering of vertices determines the numbering of the first grid points.

Figure 10: Unit cube covered by collection of simplices and the numbering of the grid points at the vertices.

In Figure 10(a) the chosen numbering results in the following additional overlap between grid points:

$$\Delta_{(0,2,0)}(\sigma^1) = \Delta_{(0,2,0)}(\sigma^4) \qquad \Delta_{(0,0,2)}(\sigma^1) = \Delta_{(0,2,0)}(\sigma^2) \tag{46}$$

$$\Delta_{(0,0,2)}(\sigma^4) = \Delta_{(0,2,0)}(\sigma^3) \qquad \Delta_{(0,0,2)}(\sigma^3) = \Delta_{(0,0,2)}(\sigma^2) \tag{47}$$

$$\Delta_{(1,1,0)}(\sigma^1) = \Delta_{(1,1,0)}(\sigma^4) \qquad \Delta_{(1,0,1)}(\sigma^1) = \Delta_{(1,1,0)}(\sigma^2) \tag{48}$$

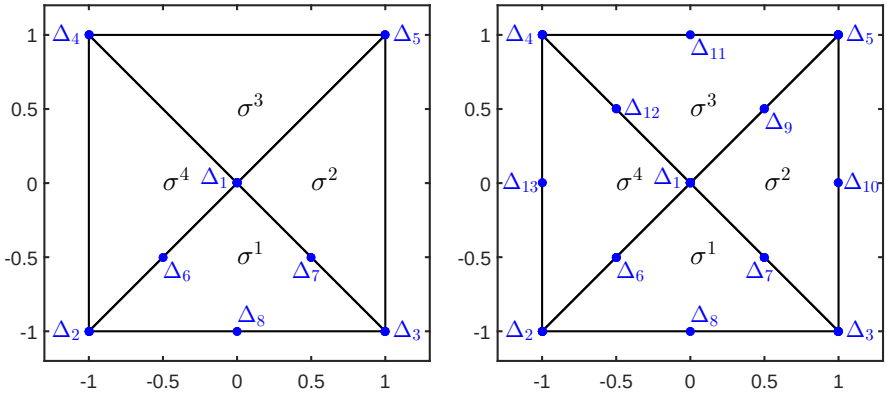
$$\Delta_{(1,0,1)}(\sigma^2) = \Delta_{(1,0,1)}(\sigma^3) \qquad \Delta_{(1,1,0)}(\sigma^3) = \Delta_{(1,0,1)}(\sigma^4). \tag{49}$$

Thus for $n = 2$, $D = 2$, and $m = 4$ there are already 11 redundant grid points out of a total number of $N_D m = 24$ grid points. Remember that N_D is the number of grid points in one simplex and m is the number of simplices in the collection. This effect gets more pronounced with higher dimension and degree.

The convention to label the unique grid points in a collection of simplices is as follows. By abuse of notation, let Δ be a list of all unique grid points. First, the numbering of the vertices carries over to the numbering of the grid points at the vertices. For Figure 10(a) this gives

$$\Delta = [\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5], \tag{50}$$

as shown on Figure 10(b). Next, run through the simplices $i = 1 \to m$ and append $\Delta_\alpha(\sigma^i)$ to Δ if the point has not already been defined. In a given simplex, the order on the α's is by the Lexicographic order imposed by Equation (16).



(a) Unit cube covered by collection of simplices and the numbering of the grid points in the first simplex.

(b) Unit cube covered by collection of simplices and the numbering of all grid points.

Figure 11: Example of the convention for labelling grid points in a collection of simplices.

In this example for σ^1, $\Delta_{(2,0,0)}(\sigma^1)$ is already Δ_1. Grid point $\Delta_{(1,1,0)}(\sigma^1)$ is not defined and thus appended as Δ_6 as seen in Figure 11(a). Next, $\Delta_{(1,0,1)}(\sigma^1)$ is also not defined and is appended as Δ_7. Continuing this way, $\Delta_{(0,2,0)}(\sigma^1)$ is already Δ_2, $\Delta_{(0,1,1)}(\sigma^1)$ is appended as Δ_8 and $\Delta_{(0,0,2)}(\sigma^1)$ is already Δ_3. Moving to σ^2, $\Delta_{(2,0,0)}(\sigma^2)$ is already Δ_1 and $\Delta_{(1,1,0)}(\sigma^2)$ is already Δ_7 as defined by $\Delta_{(1,0,1)}(\sigma^1)$. Grid point $\Delta_{(1,0,1)}(\sigma^2)$ is not defined and appended as Δ_9, and $\Delta_{(0,2,0)}(\sigma^2)$ was already Δ_3. Finally, $\Delta_{(0,1,1)}(\sigma^2)$ is appended as Δ_{10}, and $\Delta_{(0,0,2)}(\sigma^2)$ was already Δ_5. This is shown in Figure 11(b) where the process of labelling the unique grid points has been carried out for all four simplices.

This leaves

$$\Delta = [\Delta_1, \Delta_2, \ldots, \Delta_{13}], \tag{51}$$

and by denoting grid points in σ^i by Δ^i

$$\Delta^1 = [\Delta_1, \Delta_6, \Delta_7, \Delta_2, \Delta_8, \Delta_3], \qquad \Delta^2 = [\Delta_1, \Delta_7, \Delta_9, \Delta_3, \Delta_{10}, \Delta_5], \tag{52}$$

$$\Delta^3 = [\Delta_1, \Delta_{11}, \Delta_9, \Delta_4, \Delta_{12}, \Delta_5], \qquad \Delta^4 = [\Delta_1, \Delta_6, \Delta_{11}, \Delta_2, \Delta_{13}, \Delta_4], \tag{53}$$

where the ordering from Equation (16) is enforced.

I am now ready to define polynomials on collections of simplices as follows.

Definition 5 *(Polynomials on Collection of Simplices) A polynomial p of degree d in n variables is described on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ in the Bernstein basis of degree $D \geq d$ as*

$$p(x) = \begin{cases} \sum_{|\alpha|=D} b_\alpha(p, D, \sigma^1)\mathcal{B}_\alpha^D(\sigma^1) = b(p, D, \sigma^1)\mathcal{B}^D(\sigma^1) & \forall x \in \sigma^1 \\ \sum_{|\alpha|=D} b_\alpha(p, D, \sigma^2)\mathcal{B}_\alpha^D(\sigma^2) = b(p, D, \sigma^2)\mathcal{B}^D(\sigma^2) & \forall x \in \sigma^2 \\ \cdots & \\ \sum_{|\alpha|=D} b_\alpha(p, D, \sigma^m)\mathcal{B}_\alpha^D(\sigma^m) = b(p, D, \sigma^m)\mathcal{B}^D(\sigma^m) & \forall x \in \sigma^m \end{cases}, \tag{54}$$

with

$$b^T(p, D, \sigma^i) = {}^iB_j b^T(p, D, \sigma^j) \quad \forall\, i, j \in \{1, \ldots, m\}. \tag{55}$$

Similarly to the grid points, the Bernstein coefficients located on faces of the simplices in the collection will be duplicates of each other since they define the same polynomial at the same point in space. That is, the control points abide equality relations in the same manner as the grid points do. This was also seen in the example above, regarding the basis transformation matrix. Seeing as each grid point corresponds to one coefficient, the convention for labelling grid points can conveniently be adapted to label coefficients as well.

Let C denote all coefficients in the collection. Then coefficients on σ^i are given as

$$b(p, D, \sigma^i) = C(\Delta^i), \tag{56}$$

and I will often shorten the notation to $C^i = C(\Delta^i)$. This allows a more convenient definition.

Definition 6 *(Polynomials on Collection of Simplices with Unique Coefficients) With the conventions above, a polynomial p of degree d in n variables is described on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ in the Bernstein basis of degree $D \geq d$ as*

$$p(x) = \begin{cases} C^1\mathcal{B}^D(\sigma^1) & \forall x \in \sigma^1 \\ C^2\mathcal{B}^D(\sigma^2) & \forall x \in \sigma^2 \\ \cdots & \\ C^m\mathcal{B}^D(\sigma^m) & \forall x \in \sigma^m \end{cases}, \tag{57}$$

with

$${C^i}^T = {}^iB_j {C^j}^T \quad \forall\, i, j \in \{1, \ldots, m\}. \tag{58}$$

The Bernstein basis offers a natural way of extending the class of functions from polynomials to continuous piecewise-polynomials.

Definition 7 *(Continuous Piecewise-Polynomial on Collection of Simplices) With the conventions above, a continuous piecewise-polynomial p of degree d in n variables is described on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ in the Bernstein basis of degree $D \geq d$ as*

$$p(x) = \begin{cases} p_1(x) = \sum_{|\alpha|=D} b_\alpha(p_1, D, \sigma^1)\mathcal{B}^D_\alpha(\sigma^1) = b(p_1, D, \sigma^1)\mathcal{B}^D(\sigma^1) & \forall x \in \sigma^1 \\ p_2(x) = \sum_{|\alpha|=D} b_\alpha(p_2, D, \sigma^2)\mathcal{B}^D_\alpha(\sigma^2) = b(p_2, D, \sigma^2)\mathcal{B}^D(\sigma^2) & \forall x \in \sigma^2 \\ \ldots & \\ p_m(x) = \sum_{|\alpha|=D} b_\alpha(p_m, D, \sigma^m)\mathcal{B}^D_\alpha(\sigma^m) = b(p_m, D, \sigma^m)\mathcal{B}^D(\sigma^m) & \forall x \in \sigma^m \end{cases}, \tag{59}$$

with

$$\partial_k b(p_i, D, \sigma^i) = \partial_l b(p_j, D, \sigma^j) \quad \forall \left\{(k, i, l, j) | \partial_k \sigma^i = \partial_l \sigma^j\right\}. \tag{60}$$

All coefficients in the collection are denoted C, and coefficients on σ^i are identified as $b(p_i, D, \sigma^i) = C^i$.

Here the p_i's are each a polynomial on σ^i, and by the constraint Equation (60) they are continuous across the facets. The degree in the Bernstein representation of p_i is intentionally equal to D, regardless of the actual degree of p_i. This is to simplify both the representation and the constraint Equation (60). Throughout the thesis, continuous piecewise-polynomials defined on collections of simplices will always be described in the Bernstein basis of degree D on all simplices in the collection, such that

$$D \geq \max_{i \in \{1,\ldots,m\}} d_i, \tag{61}$$

where d_i is the actual degree of p_i.

In Part II, I will need another type of polynomial, the discontinuous piecewise-polynomials.

Definition 8 *(Discontinuous Piecewise-Polynomial on Collection of Simplices) With the conventions above, a discontinuous piecewise-polynomial p of degree d in n variables is described on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ in the Bernstein basis of degree $D \geq d$ as*

$$p(x) = \begin{cases} p_1(x) = \sum_{|\alpha|=D} b_\alpha(p_1, D, \sigma^1)\mathcal{B}^D_\alpha(\sigma^1) = b(p_1, D, \sigma^1)\mathcal{B}^D(\sigma^1) & \forall x \in \sigma^1 \\ p_2(x) = \sum_{|\alpha|=D} b_\alpha(p_2, D, \sigma^2)\mathcal{B}^D_\alpha(\sigma^2) = b(p_2, D, \sigma^2)\mathcal{B}^D(\sigma^2) & \forall x \in \sigma^2 \\ \ldots & \\ p_m(x) = \sum_{|\alpha|=D} b_\alpha(p_m, D, \sigma^m)\mathcal{B}^D_\alpha(\sigma^m) = b(p_m, D, \sigma^m)\mathcal{B}^D(\sigma^m) & \forall x \in \sigma^m \end{cases}. \tag{62}$$

Here the p_i's are each a polynomial on σ^i, and the lack of a constraint equation allows for them to be discontinuous across the facets. Thus there is nothing which ties (part of) $b(p_i, D, \sigma^i)$ to (part of) $b(p_j, D, \sigma^j)$. This also changes the convention about collecting all coefficients on the collection and then mapping them to different simplices. This cannot be done when simplices sharing a face do not share coefficients on said face. Instead, discontinuous piecewise-polynomials have a coefficient vector for each simplex. This makes discontinuous piecewise-polynomials set-valued on the facets.

The interest in (dis)continuous piecewise-polynomials will become evident in Part II once I tie the Bernstein basis to stability analysis. Except for some of the examples below, the rest of Part I does not consider (dis)continuous piecewise-polynomials.

Example

Consider the polynomial

$$\begin{aligned} p(x) =& 0.875x_1^3 - 10.125x_1^2x_2 + 7.5x_1^2 - 1.875x_1x_2^2 + \\ & 3x_1x_2 + 7.5x_1 + 9.125x_2^3 - 4.5x_2^2 + 10.5x_2. \end{aligned} \tag{63}$$

which is shown in Figure 12 on the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$ with the same vertices and numbering as the collection shown in Figure 11. The graph of p is the coloured surface, the solid lines are the facets of the simplices and the red, blue, and green crosses are the control points. The colour of the crosses indicate whether the coefficient is negative (red), positive (green), or equal to zero (blue).

Describing p in the Bernstein basis of degree 3 results in the 25 coefficients shown in Figure 12, and they are given as

$$\begin{aligned} C = [&0, -10, -3, 3, 22, -6, -1, -10, -11, -2, -30, \\ &-30, 6, 9, 14, 26, 10, 1, 2, 3, 14, 18, -1, 10, -10]. \end{aligned} \tag{64}$$

By the convention for numbering, the coefficients belong to the different simplices as

$$b(p, 3, \sigma^1) = C^1 = [0, -6, -1, -10, -11, -2, -10, -30, -30, -3] \tag{65}$$

$$b(p, 3, \sigma^2) = C^2 = [0, -1, 6, -2, 9, 16, -3, 26, 10, 22] \tag{66}$$

$$b(p, 3, \sigma^3) = C^3 = [0, 1, 6, 2, 3, 14, 3, 14, 18, 22] \tag{67}$$

$$b(p, 3, \sigma^4) = C^4 = [0, -6, 1, -10, -1, 2, -10, 10, -10, 3]. \tag{68}$$

■

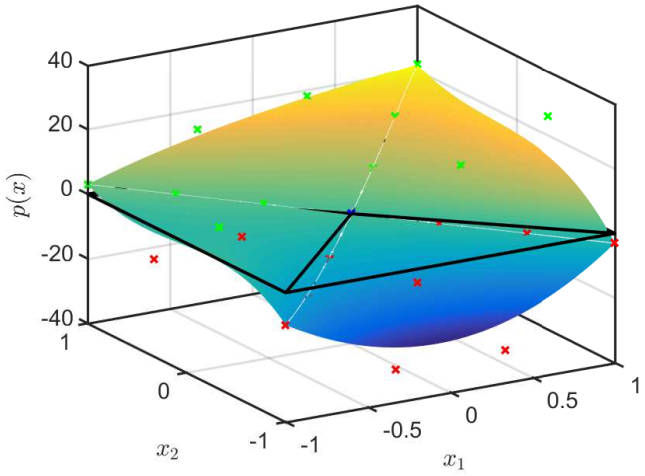


Figure 12: Polynomial on collection of simplices.

Example

Consider the continuous piecewise-polynomial

$$p(x) = \begin{cases} p_1(x) & \forall x \in \sigma^1 \\ p_2(x) & \forall x \in \sigma^2 \\ p_3(x) & \forall x \in \sigma^3 \\ p_4(x) & \forall x \in \sigma^4 \end{cases}, \tag{69}$$

with

$$\begin{aligned} p_1(x) =& 6.375x_1^3 - 1.875x_1^2x_2 - 6.75x_1^2 - 9.375x_1x_2^2 + \\ & 1.5x_1x_2 + 9x_1 + 2.875x_2^3 + 11.25x_2^2 + 9x_2 \end{aligned} \tag{70}$$

$$\begin{aligned} p_2(x) =& 11.125x_1^3 - 4.125x_1^2x_2 - 14.25x_1^2 - 13.125x_1x_2^2 - \\ & 1.5x_1x_2 + 3x_1 + 6.125x_2^3 + 15.75x_2^2 + 3x_2 \end{aligned} \tag{71}$$

$$p_3(x) = 1.5x_1 + 4.5x_2 \tag{72}$$

$$\begin{aligned} p_4(x) =& 0.875x_1^3 - 10.125x_1^2x_2 + 7.5x_1^2 - 1.875x_1x_2^2 + \\ & 3x_1x_2 + 7.5x_1 + 9.125x_2^3 - 4.5x_2^2 + 10.5x_2, \end{aligned} \tag{73}$$

which is shown in Figure 13 on the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$ with the same vertices and numbering as the collection shown in Figure 11.

The fact that p is continuous across facets is not evident from the description in the monomial basis, but in the Bernstein basis it is directly evident

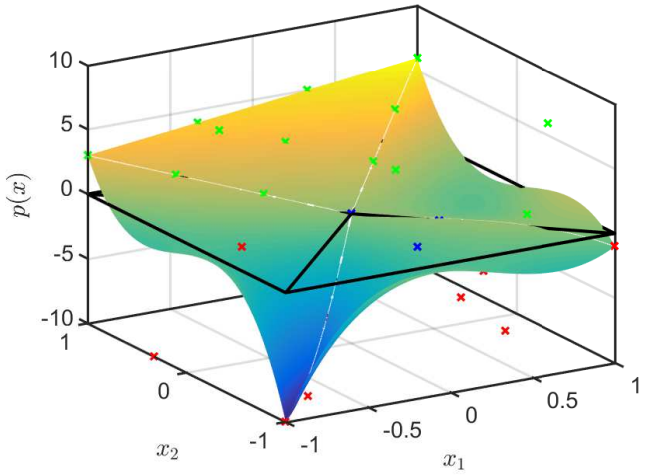


Figure 13: Continuous piecewise-polynomial on collection of simplices.

from the equality of coefficients on the facets. (Obviously, that is how I designed the example in the first place.) In the Bernstein basis of degree 3 the 25 coefficients are

$$\begin{aligned} C = [&0, -10, -1, 3, 6, -6, 0, -10, 0, 1, 8, -6, 2, \\ &-8, 4, 6, -8, 1, 2, 3, 4, 5, -1, 10, -10]. \end{aligned} \tag{74}$$

By the convention for numbering, the coefficients belong to the different simplices as

$$b(p_1, 3, \sigma^1) = C^1 = [0, -6, 0, -10, 0, 1, -10, 8, -6, -1] \tag{75}$$
$$b(p_2, 3, \sigma^2) = C^2 = [0, 0, 2, 1, -8, 4, -1, 6, -8, 6] \tag{76}$$
$$b(p_3, 3, \sigma^3) = C^3 = [0, 1, 2, 2, 3, 4, 3, 4, 5, 6] \tag{77}$$
$$b(p_4, 3, \sigma^4) = C^4 = [0, -6, 1, -10, -1, 2, -10, 10, -10, 3]. \tag{78}$$

Note that the actual degree of p_3 is 1, but since it is described in the Bernstein basis of degree 3 it has the same number of coefficients as p_2 and p_4, which makes the fulfilment of Equation (60) on the facets $\partial_1\sigma^3$ and $\partial_2\sigma^3$ straightforward. On σ^1 the coefficients on the shared facets are given as

$$\begin{aligned} \partial_2 C^1 =& [C_1, C_6, C_8, C_2] = \partial_2 b(p_1, 3, \sigma^1) \\ =& [b_{(3,0,0)}(p_1, 3, \sigma^1), b_{(2,1,0)}(p_1, 3, \sigma^1), b_{(1,2,0)}(p_1, 3, \sigma^1), b_{(0,3,0)}(p_1, 3, \sigma^1)] \\ =& [0, -6, -10, -10], \end{aligned}$$

and

$$\begin{aligned}\partial_1 C^1 =& [C_1, C_7, C_{10}, C_3] = \partial_1 b(p_1, 3, \sigma^1)\\ =& [b_{(3,0,0)}(p_1, 3, \sigma^1), b_{(2,0,1)}(p_1, 3, \sigma^1), b_{(1,0,2)}(p_1, 3, \sigma^1), b_{(0,0,3)}(p_1, 3, \sigma^1)]\\ =& [0, 0, 1, -1].\end{aligned}$$

On σ^2 the coefficients on the shared facets are given as

$$\begin{aligned}\partial_2 C^2 =& [C_1, C_7, C_{10}, C_3] = \partial_2 b(p_2, 3, \sigma^2)\\ =& [b_{(3,0,0)}(p_2, 3, \sigma^2), b_{(2,1,0)}(p_2, 3, \sigma^2), b_{(1,2,0)}(p_2, 3, \sigma^2), b_{(0,3,0)}(p_2, 3, \sigma^2)]\\ =& [0, 0, 1, -1],\end{aligned}$$

and

$$\begin{aligned}\partial_1 C^2 =& [C_1, C_{13}, C_{15}, C_5] = \partial_1 b(p_2, 3, \sigma^2)\\ =& [b_{(3,0,0)}(p_2, 3, \sigma^2), b_{(2,0,1)}(p_2, 3, \sigma^2), b_{(1,0,2)}(p_2, 3, \sigma^2), b_{(0,0,3)}(p_2, 3, \sigma^2)]\\ =& [0, 2, 4, 6].\end{aligned}$$

On σ^3 the coefficients on the shared facets are given as

$$\begin{aligned}\partial_1 C^3 =& [C_1, C_{13}, C_{15}, C_5] = \partial_1 b(p_3, 3, \sigma^3)\\ =& [b_{(3,0,0)}(p_3, 3, \sigma^3), b_{(2,0,1)}(p_3, 3, \sigma^3), b_{(1,0,2)}(p_3, 3, \sigma^3), b_{(0,0,3)}(p_3, 3, \sigma^3)]\\ =& [0, 2, 4, 6],\end{aligned}$$

and

$$\begin{aligned}\partial_2 C^3 =& [C_1, C_{18}, C_{19}, C_4] = \partial_2 b(p_3, 3, \sigma^3)\\ =& [b_{(3,0,0)}(p_3, 3, \sigma^3), b_{(2,1,0)}(p_3, 3, \sigma^3), b_{(1,2,0)}(p_3, 3, \sigma^3), b_{(0,3,0)}(p_3, 3, \sigma^3)]\\ =& [0, 1, 2, 3].\end{aligned}$$

Finally, the coefficients on the shared facets of σ^4 are given as

$$\begin{aligned}\partial_1 C^4 =& [C_1, C_{18}, C_{19}, C_4] = \partial_1 b(p_4, 3, \sigma^4)\\ =& [b_{(3,0,0)}(p_4, 3, \sigma^4), b_{(2,0,1)}(p_4, 3, \sigma^4), b_{(1,0,2)}(p_4, 3, \sigma^4), b_{(0,0,3)}(p_4, 3, \sigma^4)]\\ =& [0, 1, 2, 3],\end{aligned}$$

and

$$\begin{aligned}\partial_2 C^4 =& [C_1, C_6, C_8, C_4] = \partial_2 b(p_4, 3, \sigma^4)\\ =& [b_{(3,0,0)}(p_4, 3, \sigma^4), b_{(2,1,0)}(p_4, 3, \sigma^4), b_{(1,2,0)}(p_4, 3, \sigma^4), b_{(0,3,0)}(p_4, 3, \sigma^4)]\\ =& [0, -6, -10, -10].\end{aligned}$$

■

Example

Consider the discontinuous piecewise-polynomial

$$p(x) = \begin{cases} p_1(x) & \forall x \in \sigma^1 \\ p_2(x) & \forall x \in \sigma^2 \\ p_3(x) & \forall x \in \sigma^3 \\ p_4(x) & \forall x \in \sigma^4 \end{cases}, \tag{79}$$

with

$$\begin{aligned} p_1(x) =& -1.5x_1^3 + 1.5x_1^2x_2 + 0.75x_1^2 + 16.5x_1x_2^2 + 31.5x_1x_2 \\ &+ 12x_1 + 18.5x_2^3 + 24.75x_2^2 + 3x_2 \end{aligned} \tag{80}$$

$$\begin{aligned} p_2(x) =& 48.875x_1^3 - 19.875x_1^2x_2 - 76.5x_1^2 - 6.375x_1x_2^2 + 15x_1x_2 + \\ &31.5x_1 + 1.375x_2^3 + 4.5x_2^2 + 1.5x_2 \end{aligned} \tag{81}$$

$$p_3(x) = 3.875x_1^3 + 1.875x_1^2x_2 - 0.375x_1x_2^2 + 1.625x_2^3 \tag{82}$$

$$\begin{aligned} p_4(x) =& -4.5x_1^3 - 12x_1^2x_2 + 1.5x_1^2 - 7.5x_1x_2^2 \\ &- 15x_1x_2 + 6x_1 - 7.5x_2^2 - 3x_2, \end{aligned} \tag{83}$$

which is shown in Figure 14 on the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$ with the same vertices and numbering as the collection shown in Figure 11.

The fact that p is discontinuous across facets is not evident from the description in the monomial basis, but in the Bernstein basis it is directly evident from the coefficients on the facets. In the Bernstein basis of degree 3, the coefficients are

$$b(p_1, 3, \sigma^1) = [0, -5, 3, 9, 6, 4, 7, 3, 4, -2] \tag{84}$$

$$b(p_2, 3, \sigma^2) = [0, 10, 11, -9, -6, 3, 4, 7, 2, 0] \tag{85}$$

$$b(p_3, 3, \sigma^3) = [0, 0, 0, 0, 0, 0, 0, 5, -3, 7] \tag{86}$$

$$b(p_4, 3, \sigma^4) = [0, -1, -3, -9, -1, -3, 0, 0, 0, 0]. \tag{87}$$

On σ^1 the coefficients on the shared facets are given as

$$\partial_2 b(p_1, 3, \sigma^1) = [0, -5, 9, 7] \qquad \partial_1 b(p_1, 3, \sigma^1) = [0, 3, 4, -2]. \tag{88}$$

On σ^2 the coefficients on the shared facets are given as

$$\partial_2 b(p_2, 3, \sigma^2) = [0, 10, -9, 4], \qquad \partial_1 b(p_2, 3, \sigma^2) = [0, 11, 3, 0]. \tag{89}$$

On σ^3 the coefficients on the shared facets are given as

$$\partial_1 b(p_3, 3, \sigma^3) = [0, 0, 0, 7], \qquad \partial_2 b(p_3, 3, \sigma^3) = [0, 0, 0, 0]. \tag{90}$$

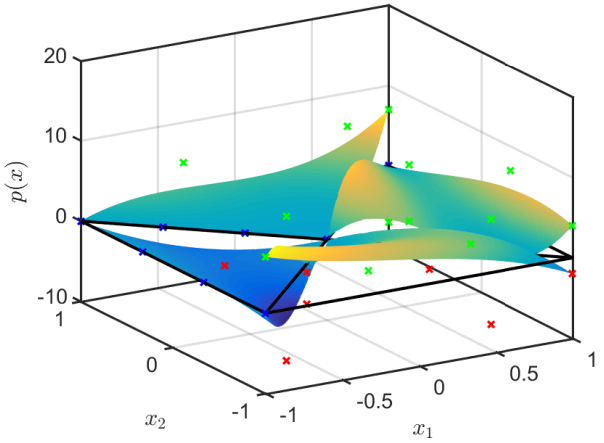


Figure 14: Discontinuous piecewise-polynomial on collection of simplices.

The coefficients on the shared facets of σ^4 are given as

$$\partial_1 b(p_4, 3, \sigma^4) = [0, -3, -3, 0], \qquad \partial_2 b(p_4, 3, \sigma^4) = [0, -1, -9, 0]. \tag{91}$$

■

This concludes the material presented in this Chapter. The next section will cover the implementation of all the mathematical operations presented in the above; how to define polynomials in the Bernstein basis directly on collections of simplices, how to transform between descriptions on different simplices, how to convert from monomial basis to Bernstein basis, how to obtain a triangulation, and more.

4 Software

In this section, I will cover the implementation of the operations presented in this Chapter. The present software section will be significantly longer than the following software sections, since many basic operation are covered. I will start out by introducing the structure of how all following functions handle collections of simplices.

To construct a collection of simplices, two matrices are needed. One named *vex* which contains all the vertices in the collection and one named *simplex* which contains the combinations of vertices to define the simplices. As an example, the collection of simplices in Figure 10(a) is defined by

$$vex = \begin{bmatrix} 0 & -1 & 1 & -1 & 1 \\ 0 & -1 & -1 & 1 & 1 \end{bmatrix}, \qquad simplex = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \\ 1 & 4 & 5 \\ 1 & 2 & 4 \end{bmatrix}, \tag{92}$$

such that e.g. σ^3 is identified as

$$vex(:, simplex(3,:)) = \begin{bmatrix} 0 & -1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{93}$$

with $\sigma_0^3 = [0,0]^T$, $\sigma_1^3 = [-1,1]^T$, and $\sigma_2^3 = [1,1]^T$.

The function *getBarycentricCoordinates* takes one simplex as an input and returns the barycentric coordinates for the simplex, according to Equation (4).

The function *getInitialPartition* takes an $n \times 2$ matrix with minimum and maximum values in each variable defining an arbitrary n-dimensional box, and returns *vex* and *simplex*. This is done by performing Kuhn's triangulation on each facet by running through all the possible permutations of vertices on each facet and creating initial simplices. Adding the origin to *vex* and shifting the index of all vertices in *simplex* by one, the desired collection is obtained. The collection covers the box, and if the origin is not in the interior of the box, the function returns an *error*.

The function *getSimplexSorted* takes one input, a matrix *simplex*, and returns the same matrix after sorting the rows such that the simplices are ordered according to the convention that $\forall\ i \in \{1, \ldots, m-1\}$ simplices σ^i and σ^{i+1} overlaps with one facet.

The function *genpow* takes two inputs, n and D, and returns the α-matrix of different α-combinations according to Equation (16). *genpow* is part of the GloptiPoly toolbox for polynomial manipulation and optimisation using the moment approach, see [19].

The function *binomCoef* takes two inputs, a scalar D and vector *alpha*, and returns the binomial coefficient according to Equation (14).

The function *getBernsteinBasisPolynomials* takes two inputs, D and vex, and returns the Bernstein basis polynomials of degree D on the simplex defined by the vertices in vex according to Equation (15).

The function *getMon2BernSimplex* is used to transform the description of a polynomial in the monomial basis to a description in the Bernstein basis on one simplex σ. *getMon2BernSimplex* takes two inputs, $gamma$ and vex, where $gamma$ is the exponent matrix in Equation (12) and vex is a matrix with vertices defining the simplex. It returns a matrix $M2B$, as derived in Appendix B, the α-matrix, and the degree d of the polynomial. The matrix $M2B$ can then be used to transform the coefficients in the monomial basis to the coefficients in the Bernstein basis by

$$b^T(p, d, \sigma) = M2B\ c^T \tag{94}$$

where the ordering of the rows in $M2B$ is in accordance with the convention of the numbering of the vertices in vex. This ensures that

$$b(p, d, \sigma) = [b_{(d,0,\ldots,0)}(p, d, \sigma), b_{(d-1,1,\ldots,0)}(p, d, \sigma), \ldots, b_{(0,0,\ldots,d)}(p, d, \sigma)]. \tag{95}$$

Note that *getMon2BernSimplex* utilises two functions, *getProduct* and *elevateDegree*, which have not been introduced yet. They are introduced in Section 9 after multiplication and degree elevation have been introduced.

The function *setCoef* is used to alter an entry in a vector. It takes the inputs n, d, C, $alpha$, and $Cval$. They are the number of variables, the degree, the vector, the α-combination, and the new value respectively. It uses the function *getCoefIndex* which returns the index of a given α-combination. The output of *setCoef* is the vector C, now with the new value.

The function *getMon2BernTrans* is used to transform the description of a polynomial in the monomial basis to a description in the Bernstein basis on a collection of simplices K. *getMon2BernTrans* takes four inputs, n and the exponent matrix $gamma$ form Equation (12), as well as vex and $simplex$ defining the collection. First, all control points in the collection are determined according to the convention presented in Section 3, using the function *getControlPoints*. *getControlPoints* also bookkeeps the maps Δ^i as shown in Equations (52) and (53) in the matrix $simplexCtrlPointsVF$. Then the matrix $M2B$ is created and the first rows are defined using *getMon2BernSimplex*. Finally, the remaining rows in $M2B$ are calculated similarly to how it is done in *getMon2BernSimplex*, but with the ordering given by all control points in the collection. *getMon2BernTrans* returns the matrix $M2B$, the α-matrix, the degree d of the polynomial, and the matrix $simplexCtrlPointsVF$. The Bernstein basis coefficients on the collection can then be calculated by

$$C^T = M2B\ c^T \tag{96}$$

and the coefficients on simplex σ^i are given by

$$b(p, d, \sigma^i) = C^i = C(simplexCtrlPointsVF(i, :)). \tag{97}$$

The function returns an *error* if the numbering of the simplices is not done in accordance with the convention from above.

The function *getPoly* takes five inputs, b, n, d, $alpha$, and vex, and returns a polynomial p in the monomial basis. b is the Bernstein basis coefficients of p in degree d on the simplex defined by the vertices in vex. $alpha$ is the α-matrix.

The function *getCtrlPointBernTrans* is used to obtain the matrix for transforming between descriptions of a polynomial on different simplices. It takes four inputs; n, d, and the collection defined by vex and $simplex$. First, all control points in the collection are determined according to the convention presented in Section 3, using the function *getControlPoints*. Then the matrix $B2C$ is created as derived in Appendix B and the last simplex in the collection, σ^m, is assumed to be the simplex on which p is initially described. The remaining rows in $B2C$ are calculated similarly to how it is done in *getMon2BernSimplex*, but with the ordering given by all control points in the collection. *getCtrlPointBernTrans* returns the matrix $B2C$, the α-matrix, the degree d of the polynomial, and the matrix $simplexCtrlPointsVF$. The Bernstein basis coefficients on the collection can then be calculated from the Bernstein basis coefficients on σ^m by

$$C^T = B2C\ b^T(p, D, \sigma^m) \tag{98}$$

and the coefficients on simplex σ^i are given by

$$b(p, d, \sigma^i) = C^i = C(simplexCtrlPointsVF(i, :)). \tag{99}$$

The function returns an *error* if the numbering of the simplices is not done in accordance with the convention from above.

The function *plotCollection* can be used for an easy visualisation of a two or three dimensional collection of simplices. The inputs are the collection vex and $simplex$ and produces a figure, there is no output.

Bernstein Basis Properties

The Bernstein basis offers several unique properties which have made the basis interesting for various reasons. In this Chapter, I introduce the properties which are relevant for later use in stability analysis. The properties are from [14] and the interested reader can consult [14] for a thorough treatment of all properties of the basis.

To aid the presentation of the properties, Figure 15 shows the polynomial

$$p(x) = 10x^3 - 27x^2 + 18x + 5 \tag{100}$$

on the collection of simplices $K = \{\sigma^1, \sigma^2\}$. At the bottom of the figure, the simplices σ^i, the vertices σ^i_j (black diamonds), and the grid points $\Delta_\alpha(\sigma^i)$ (blue circles) are shown. The graph of p is shown in red and the control points $(\Delta_\alpha(\sigma^i), b_\alpha(\sigma^i))$ are shown using black crosses where the dependency on σ^i has been omitted from the figure. These quantities were all introduced in the previous chapter.

The solid lines in black outline the convex hull of the control points on each simplex and exemplify the convex hull property.

Convex Hull Property:

On a given simplex, the graph of a polynomial is contained in the convex hull of its control points when described in the Bernstein basis on the simplex.

On σ^i, all control points together constitute the control net of p on σ^i. The so-called discrete graph of p is shown using blue stars. The discrete graph consists of the points

$$(\Delta_\alpha(\sigma^i), p(\Delta_\alpha(\sigma^i))), \tag{101}$$

and the dependency on σ^i has been omitted from the figure. Also, only some of the discrete graph is labelled due to clarity. In particular,

$$(\Delta_{(3,0)}(\sigma^1), p(\Delta_{(3,0)}(\sigma^1))), \qquad (\Delta_{(0,3)}(\sigma^1), p(\Delta_{(0,3)}(\sigma^1))), \tag{102}$$

$$(\Delta_{(3,0)}(\sigma^2), p(\Delta_{(3,0)}(\sigma^2))), \qquad (\Delta_{(2,1)}(\sigma^2), p(\Delta_{(2,1)}(\sigma^2))), \tag{103}$$

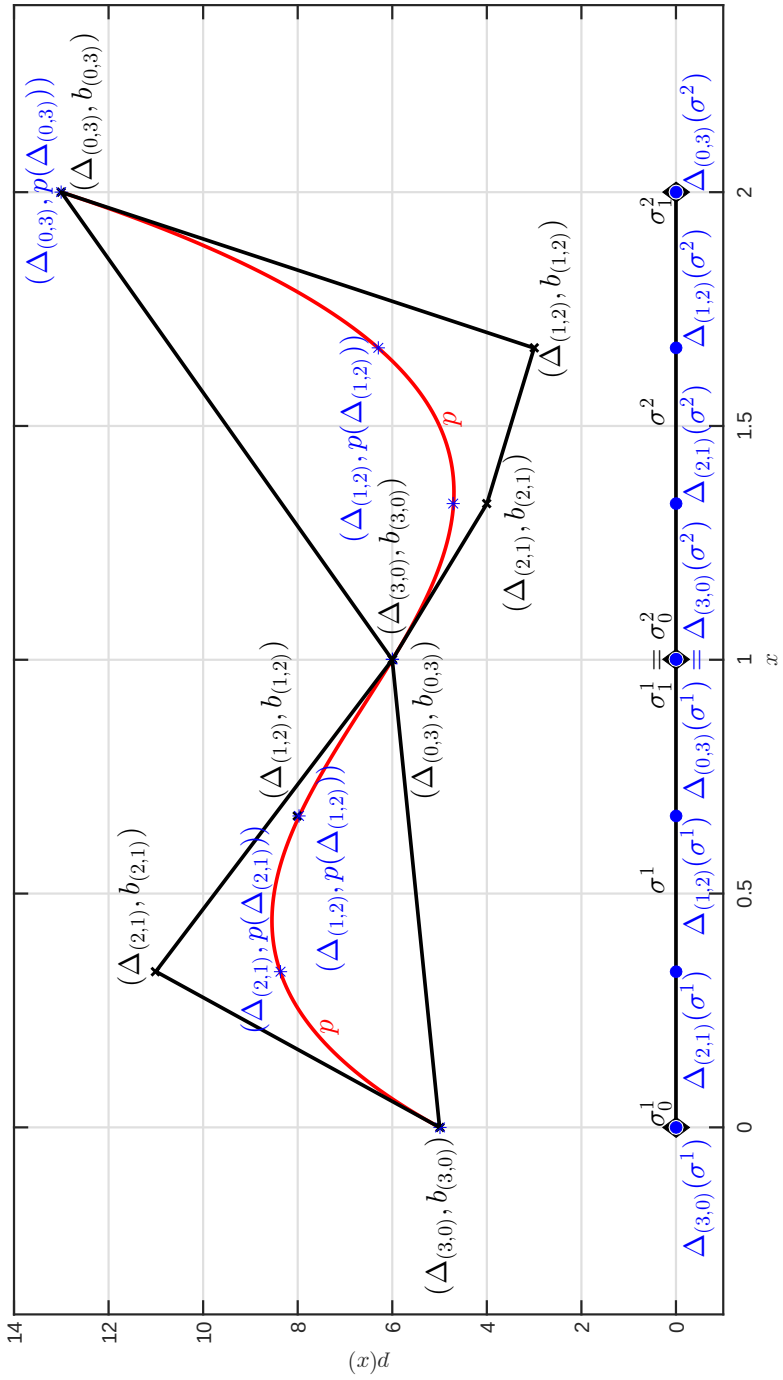


Figure 15: Graph of p, vertices, grid points, control points, convex hull, and discrete graph of p.

are missing on the figure. Each point in the discrete graph is placed on the (continuous) graph of p. In contrast, only some of the points in the control net are on the (continuous) graph. This happens for the grid points at the vertices, and this is in fact the result of the end-point value property.

End-Point Value Property:

The value of a polynomial p of degree d when evaluated at a vertex is equal to the coefficient at the vertex. Thus on any simplex σ, for $j \in \{0, 1, \ldots, n\}$ the following relation holds:

$$p\left(\Delta_{(De_j)}(D, \sigma)\right) = b_{(De_j)}(p, D, \sigma), \tag{104}$$

where $D \geq d$.

In [26] it is shown that the control net converges to the discrete graph under either degree elevation or sub-division.

Degree elevation is the process of converting the description of a polynomial p of degree d in the Bernstein basis of degree $D_1 \geq d$ to a description in the Bernstein basis of degree $D_2 > D_1$. This procedure is covered in Chapter Arithmetic. For now, note that the number of control points grows when the degree of the description is elevated and that the gap between the control net and the discrete graph diminishes, and for $D \to \infty$ vanishes.

Sub-division is the process of splitting a simplex into a collection of m (smaller) simplices. This procedure is covered in Chapter Positivity Certification. In a manner similar to degree elevation, this increases the number of control points and the gap between the control net and the discrete graph diminishes, and for $m \to \infty$ vanishes. It will later become of great importance how to split simplices into smaller simplices.

Non-Negativity:

Directly evident from the definition of the basis polynomials, Equation (15), is the non-negativity of $\mathcal{B}(\sigma)$ for $x \in \sigma$.

A direct result of the non-negativity of the basis polynomials is that the sign of the Bernstein coefficients $b_\alpha(p, D, \sigma)$ of a polynomial p can determine the sign of p on σ. If all coefficients are non-negative, then so is p. The same holds for positivity, non-positivity, and negativity.

In the same line of thought, the convex hull property yields (crude) lower and upper bounds on p.

Lower And Upper Bounds:

For a polynomial p, for $x \in \sigma$ the following relation holds

$$\min_{|\alpha|=D} b_\alpha \leq p(x) \leq \max_{|\alpha|=D} b_\alpha, \tag{105}$$

where equality only happens if either extrema occurs for $\alpha = De_j$ by the end-point value property. This is the case on Figure 15 for the minimum of p on σ^1 equal to $b_{(3,0)}$ and the maximum of p on σ^2 equal to $b_{(0,3)}$.

The last property of interest is the fact that the basis polynomials are unimodal.

Unimodality:

For $x \in \sigma$, the basis polynomial $\mathcal{B}^D_{\bar{\alpha}}(\sigma)$ has a unique maximum at $\Delta_{\bar{\alpha}}(D, \sigma)$. Also, for $x = \Delta_{\bar{\alpha}}$ the following relation holds

$$\mathcal{B}^D_{\bar{\alpha}} > \mathcal{B}^D_{\alpha}, \ \forall \alpha \in \hat{\alpha} \tag{106}$$

where $\hat{\alpha}$ is the α-matrix with the combination $\bar{\alpha}$ removed.

A consequence of the unimodality of the basis polynomials, is that at $\Delta_{\hat{\alpha}}$ the coefficient $b_{\hat{\alpha}}$ contributes more to the graph of p than any other coefficient.

Arithmetic

This Chapter presents of a few but essential mathematical operations on polynomials described in the simplicial Bernstein basis. The results for multiplication, degree elevation, and differentiation can be found in [9]. The result for addition can be found in e.g. [15] although presented for the tensorial Bernstein basis. The material presented in this Chapter includes thorough derivations behind the results. The implementation of the operations is covered in the final section of this Chapter.

Throughout the chapter, I need two polynomials in the derivations. They are defined as

$$p_1 = \sum_{|\alpha|=D_1} b_\alpha(p_1, D_1, \sigma)\mathcal{B}_\alpha^{D_1}, \qquad p_2 = \sum_{|\beta|=D_2} b_\beta(p_2, D_2, \sigma)\mathcal{B}_\beta^{D_2}, \tag{107}$$

and to shorten the notation, I define the coefficient vector of p_1 as $b^1 = b(p_1, D_1, \sigma)$ and the coefficient vector of p_2 as $b^2 = b(p_2, D_2, \sigma)$. Note that any arithmetic performed on polynomials defined on collections of simplices, is simply performed on each simplex individually. For this reason, the dependency on σ is left out henceforth in this Chapter.

5 Multiplication

Let the polynomial h be the product of p_1 and p_2 as $h = p_1 p_2$ and write

$$h = \sum_{|\gamma|=D_h} b_\gamma(h, D_h)\mathcal{B}_\gamma^{D_h} = b^h \mathcal{B}^{D_h}. \tag{108}$$

Note that $D_h = D_1 + D_2$ and $|\gamma| = |\alpha + \beta|$.

Writing out $p_1 p_2$ and identifying the Bernstein basis of degree D_h, the

coefficients b_γ^h can be identified as

$$h = \sum_{|\alpha|=D_1} \sum_{|\beta|=D_2} b_\alpha^1 b_\beta^2 \mathcal{B}_\alpha^{D_1} \mathcal{B}_\beta^{D_2} \tag{109}$$

$$= \sum_{|\alpha|=D_1} \sum_{|\beta|=D_2} b_\alpha^1 b_\beta^2 \binom{D_1}{\alpha} \lambda^\alpha \binom{D_2}{\beta} \lambda^\beta \tag{110}$$

$$= \sum_{|\alpha|=D_1} \sum_{|\beta|=D_2} b_\alpha^1 b_\beta^2 \binom{D_1}{\alpha} \binom{D_2}{\beta} \lambda^{\alpha+\beta} \tag{111}$$

$$= \sum_{|\alpha|=D_1} \sum_{|\beta|=D_2} b_\alpha^1 b_\beta^2 \frac{\binom{D_1}{\alpha}\binom{D_2}{\beta}}{\binom{D_h}{\gamma}} \underbrace{\binom{D_h}{\gamma} \lambda^{\alpha+\beta}}_{\mathcal{B}_\gamma^{D_h}} \tag{112}$$

$$= \sum_{|\gamma|=D_h} b_\gamma^h \mathcal{B}_\gamma^{D_h} \tag{113}$$

where the coefficients of h are

$$b_\gamma^h = \sum_{\substack{|\gamma-\alpha|=D_2 \\ \gamma-\alpha \geq 0}} b_\alpha^1 b_{\gamma-\alpha}^2 \frac{\binom{D_1}{\alpha}\binom{D_2}{\gamma-\alpha}}{\binom{D_h}{\gamma}}, \qquad \forall |\gamma| = D_h \tag{114}$$

with α being the summation variable.

6 Degree Elevation

The polynomial p_1 described in the Bernstein basis of degree D_1 can be described equally well in a Bernstein basis of degree $D = D_1 + r$ where $r \in \mathbb{N}$ determines how much the degree is elevated. Thus

$$p_1 = \sum_{|\alpha|=D_1} b_\alpha(p_1, D_1) \mathcal{B}_\alpha^{D_1} = \sum_{|\gamma|=D} b_\gamma(p_1, D) \mathcal{B}_\gamma^D \tag{115}$$

where the coefficients are related as

$$b_\gamma(p_1, D) = \sum_{\substack{|\gamma-\alpha|=r \\ \gamma-\alpha \geq 0}} b_\alpha(p_1, D_1) \frac{\binom{D_1}{\alpha}\binom{r}{\gamma-\alpha}}{\binom{D}{\gamma}}, \qquad \forall |\gamma| = D \tag{116}$$

with α as the summation variable. This relation is obtained in the same way as the derivation of the multiplication, except that p_2 is fixed to the unit polynomial and described in the Bernstein basis of degree $D_2 = r$. From Equation (18) it is easily seen that the unit polynomial described in any degree is obtained with a coefficient vector b^2 of all ones.

7 Addition

Let the polynomial g be the sum of p_1 and p_2 as $g = p_1 + p_2$ and write

$$g = \sum_{|\gamma|=D_g} b_\gamma(g, D_g)\mathcal{B}_\gamma^{D_g} = b^g \mathcal{B}^{D_g}. \tag{117}$$

If $D_1 = D_2$ the degree of g is also $D_g = D_1 = D_2$ and the coefficients of g are then easily determined as

$$b_\alpha(g, D_g) = b_\alpha(p_1, D_1) + b_\alpha(p_2, D_2), \qquad \forall |\alpha| = D_g. \tag{118}$$

If e.g. $D_1 > D_2$ the degree of g is $D_g = D_1$ and the degree of p_2 must be elevated by $r = D_1 - D_2$ before the coefficients can be added.

8 Differentiation

This Section covers the derivation of how to compute the partial derivative of a polynomial defined in the Bernstein basis. This is done in three steps, where first the partial derivative of the basis polynomials with respect to the barycentric coordinates is derived. Then the partial derivative of the basis polynomials with respect to the variables is covered, and finally, the partial derivative of polynomials in the Bernstein basis with respect to the variables is derived.

Partial Derivative of Basis Polynomials with Respect to Barycentric Coordinates

By Equations (14) and (15) the basis polynomials are

$$\mathcal{B}_\alpha^D = \binom{D}{\alpha}\lambda^\alpha = \frac{D!}{\alpha_0!\alpha_1!\cdots\alpha_n!}\prod_{i=0}^{n}\lambda_i^{\alpha_i}. \tag{119}$$

Differentiating with respect to λ_j yields

$$\frac{\partial \mathcal{B}_\alpha^D}{\partial \lambda_j} = \alpha_j \frac{D!}{\alpha_0!\alpha_1!\cdots\alpha_n!}\left(\prod_{\substack{i=0\\ i\neq j}}^{n}\lambda_i^{\alpha_i}\right)\lambda_j^{\alpha_j-1} = \alpha_j \frac{D!}{\alpha_0!\alpha_1!\cdots\alpha_n!}\lambda^{\alpha-e_j}, \tag{120}$$

which can be rearranged, assuming $\alpha_j \geq 1$, into

$$\frac{\partial \mathcal{B}_\alpha^D}{\partial \lambda_j} = D\frac{(D-1)!}{\alpha_0!\cdots\alpha_{j-1}!(\alpha_j-1)!\alpha_{j+1}\cdots\alpha_n!}\lambda^{\alpha-e_j} = D\binom{D-1}{\alpha-e_j}\lambda^{\alpha-e_j} = D\mathcal{B}_{\alpha-e_j}^{D-1}. \tag{121}$$

If $\alpha_j = 0$, the partial derivative equals zero, and $\alpha_j - e_j = -1 \notin \mathbb{N}_0$. To overcome this, I use the convention

$$\mathcal{B}_\alpha^D \equiv 0, \qquad \forall \alpha \notin \mathbb{N}_0^{n+1}, \tag{122}$$

such that the results combine to

$$\frac{\partial \mathcal{B}_\alpha^D}{\partial \lambda_j} = D\mathcal{B}_{\alpha - e_j}^{D-1} \tag{123}$$

for $\alpha_j \geq 0$ and $j \in \{0, 1, \ldots, n\}$.

Partial Derivative of Basis Polynomials with Respect to Variables

By Equation (4) the barycentric coordinates are defined by

$$\begin{bmatrix} x_1 \\ \cdots \\ x_n \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \cdots & \sigma_n(1) \\ \cdots & \cdots & \cdots \\ \sigma_0(n) & \cdots & \sigma_n(n) \\ 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \cdots \\ \lambda_n \end{bmatrix} \tag{124}$$

which, when differentiated with respect to the variables, yields

$$\begin{bmatrix} I_{n\times n} \\ 0_{1\times n} \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \cdots & \sigma_n(1) \\ \cdots & \cdots & \cdots \\ \sigma_0(n) & \cdots & \sigma_n(n) \\ 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial \lambda_0}{\partial x_1} & \cdots & \frac{\partial \lambda_0}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial \lambda_n}{\partial x_1} & \cdots & \frac{\partial \lambda_n}{\partial x_n} \end{bmatrix}, \tag{125}$$

with $I_{n\times n}$ the identity matrix, and $0_{1\times n}$ a row vector of zeros. As the vertices σ_i are affinely independent, the derivative of the barycentric coordinates can be described as

$$\frac{\partial \lambda}{\partial x} = \begin{bmatrix} \frac{\partial \lambda_0}{\partial x_1} & \cdots & \frac{\partial \lambda_0}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial \lambda_n}{\partial x_1} & \cdots & \frac{\partial \lambda_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \cdots & \sigma_n(1) \\ \cdots & \cdots & \cdots \\ \sigma_0(n) & \cdots & \sigma_n(n) \\ 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} I_{n\times n} \\ 0_{1\times n} \end{bmatrix}, \tag{126}$$

and for compact notation define

$$\varsigma = \begin{bmatrix} \sigma_0(1) & \cdots & \sigma_n(1) \\ \cdots & \cdots & \cdots \\ \sigma_0(n) & \cdots & \sigma_n(n) \\ 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} I_{n\times n} \\ 0_{1\times n} \end{bmatrix}, \tag{127}$$

making $\varsigma \in \mathbb{R}^{n+1\times n}$. Finally, using the chain rule and Equation (123)

$$\frac{\partial \mathcal{B}_\alpha^D}{\partial x} = \frac{\partial \mathcal{B}_\alpha^D}{\partial \lambda} \frac{\partial \lambda}{\partial x} = D[\mathcal{B}_{\alpha - e_0}^{D-1}, \ldots, \mathcal{B}_{\alpha - e_n}^{D-1}]\varsigma. \tag{128}$$

Partial Derivative of Polynomials in the Bernstein Basis

Let a polynomial p be described by

$$p = \sum_{|\alpha|=D} b_\alpha(p, D)\mathcal{B}_\alpha^D, \tag{129}$$

and denote its derivative by

$$p' = \frac{\partial p}{\partial x} = \sum_{|\tilde{\alpha}|=D-1} b_{\tilde{\alpha}}(p', D-1)\mathcal{B}_{\tilde{\alpha}}^{D-1} \tag{130}$$

with $b_{\tilde{\alpha}}(p', D-1) \in \mathbb{R}^{1\times n}$. Then the relationship between $b_\alpha(p, D)$ and $b_{\tilde{\alpha}}(p', D-1)$ is derived as

$$\begin{aligned}
\frac{\partial p}{\partial x} &= \sum_{|\alpha|=D} b_\alpha(p, D)\frac{\partial \mathcal{B}_\alpha^D}{\partial x} && (131)\\
&= \sum_{|\alpha|=D} D b_\alpha(p, D)[\mathcal{B}_{\alpha-e_0}^{D-1}, \ldots, \mathcal{B}_{\alpha-e_n}^{D-1}]\varsigma && (132)\\
&= \sum_{|\alpha|=D} D[b_\alpha(p, D)\mathcal{B}_{\alpha-e_0}^{D-1}, \ldots, b_\alpha(p, D)\mathcal{B}_{\alpha-e_n}^{D-1}]\varsigma. && (133)
\end{aligned}$$

Defining a new summation variable by $\alpha = \tilde{\alpha} + e_j \;\forall\; j \in \{0, 1, \ldots, n\}$ gives

$$\frac{\partial p}{\partial x} = \sum_{|\tilde{\alpha}|=D-1} D[b_{\tilde{\alpha}+e_0}(p, D), \ldots, b_{\tilde{\alpha}+e_n}(p, D)]\varsigma\mathcal{B}_{\tilde{\alpha}}^{D-1} \tag{134}$$

which, by comparing to Equation (130) identifies

$$b_{\tilde{\alpha}}(p', D-1) = D[b_{\tilde{\alpha}+e_0}(p, D), \ldots, b_{\tilde{\alpha}+e_n}(p, D)]\varsigma, \qquad \forall|\tilde{\alpha}| = D-1, \tag{135}$$

with $b_{\tilde{\alpha}}(p', D-1) \in \mathbb{R}^{1\times n}$.

9 Software

The function *getProduct* takes seven inputs, defining two polynomials, and returns the product of them. Cf, $alphaf$, and df are the coefficient vector, the α-matrix, and the degree of the first polynomial and Cg, $alphag$, and dg are the coefficient vector, the α-matrix, and the degree of the second polynomial. Note that $alphag$ is actually not used in the function. Finally, n is the dimension of the polynomials. The output polynomial is defined by C, $alpha$, and d which are the coefficient vector, the α-matrix, and the degree of the product.

The function *elevateDegree* takes five inputs, defining one polynomial, and returns the representation of the same polynomial in a higher degree. r is how much the degree is to be elevated. Cd, $Alpha$, d, and n are the coefficient vector, the α-matrix, the degree, and the dimension of the polynomial, respectively. The output is Cr, $Gamma$, and dr as the coefficient vector, the α-matrix, and the degree of the elevated representation.

The function *getSum* takes seven inputs, defining two polynomials, and returns the sum of them. Cf, $alphaf$, and df are the coefficient vector, the α-matrix, and the degree of the first polynomial and Cg, $alphag$, and dg are the coefficient vector, the α-matrix, and the degree of the second polynomial, and n is the dimension. The output polynomial is defined by C, $alpha$, and d which are the coefficient vector, the α-matrix, and the degree of the sum.

Note that there is no implementation of the differentiation. In what follows, differentiation is not needed as a standalone function, but the equation is incorporated in a bigger function in the stability part. The function *getLieDerivativeTrans* utilises the above, as explained in Section 21.

Positivity Certification

With the introduction to the Bernstein basis thoroughly covered, including some properties and arithmetic, the time has now come to actually start using the basis for something. This Chapter covers a straightforward utilization of the basis for positivity certification.

In general, positivity certification is the process of certifying the positivity of a function. It can be either over the entire domain of definition or on sub-domains. This is referred to as either certifying global or local positivity.

Certifying the positivity of a polynomial can be done in many ways, but the common denominator is that the positivity is certified by the existence of some *certificate of positivity*. A certificate of positivity can be thought of as a one-line proof of positivity. It could be the existence of a certain algebraic relation, the existence of a certain matrix with positive eigenvalues, or the existence of a certain vector with positive entries.

Some of the theoretical results regarding algebraic relations are the celebrated *Stellensatze*, be it Hilbert's Nullstellensatz [39], Stengle's Positivestellensatz [49], Putinar's Positivestellensatz [25], or variations hereof. They go far beyond positivity certification, but can be used for it. Another relation is the straightforward utilising of sum-of-squares polynomials, $\Sigma[X]$, where the reasoning is that if a polynomial can be expressed as a sum of squares, it must necessarily be at least non-negative. Further inspection of the minima of the squares can assess if the polynomial is positive definite or even positive. See Appendix D for an example of this.

Certifying polynomial positivity using the Bernstein basis is obtained directly by the sign of the coefficients. This is thanks to the convex hull property and the derived lower bounds. If all coefficients are positive, then so is the polynomial. If one or more coefficients are zero, while the rest are positive, then the polynomial is either positive or non-negative, depending on which coefficients are equal to zero. This is by the end-point value property. If all coefficients are equal to zero, the polynomial is the zero polynomial, and if the sign of the coefficients are mixed, nothing can be said (directly) about the sign of the polynomial. If the coefficients at the vertices are of mixed sign, then so is the polynomial. When certifying positivity of polynomials using

the Bernstein basis, the certificate will inherently be a local certificate, due to the fact that the basis polynomials are defined on simplices.

If a polynomial is to be investigated for positivity on a given simplex and the initial description in the Bernstein basis results in a coefficient vector of mixed sign, the polynomial may still be positive. Sub-dividing into smaller simplices, or raising the degree of the representation, could result in a vector of positive coefficients. This is covered in the following, where the so-called Bernstein's Theorem constitutes the theoretical foundation for the search for a certificate of positivity.

10 By Degree Elevation

The motivation for investigating positivity of polynomials by degree elevation, is Bernstein's Theorem.

Theorem 9 *(Bernstein's Theorem: Degree Elevation [26]) Let the polynomial p, of actual degree d, be positive on the simplex σ. Then there exists a degree $D \geq d$ such that all entries of $b(p, D, \sigma)$ are positive.*

Bernstein's Theorem ensures the ability to certify the positivity of positive polynomials by degree elevation, and the (local) certificate of positivity is the coefficient vector $b(p, D, \sigma)$. A theoretical bound on the degree D is derived in [26]. Unfortunately, it relies on the minimum of the polynomial on the simplex. However, in a practical setting, it is comfortable to know that there is a bound on D.

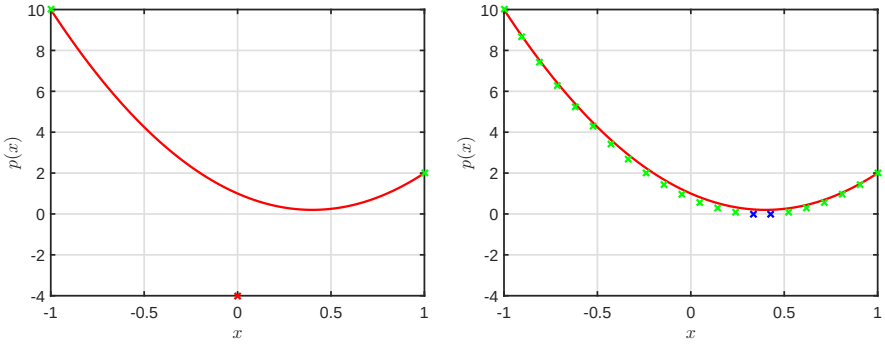
Except for the following example, certification by degree elevation will not be considered any further in this Thesis. While working on the subject, it became evident to me that the certification process using degree elevation is outperformed by the certification process using sub-division. This is in alignment with the findings of other authors, see e.g. [10].

Example

Consider the polynomial $p(x) = 5x^2 - 4x + 1$. It is wished to investigate the local positivity of p on the simplex $\sigma = [-1, 1]$. Described on σ in the Bernstein basis of degree 2, the coefficient vector is $b(p, 2, [-1, 1]) = [10, -4, 2]$. Since the coefficients vector has a negative element, $b(p, 2, [-1, 1])$ is not a certificate of positivity.

Raising the degree of the description to $D = 21$ the (rounded) coefficient vector is

$$\begin{aligned} b(p, 21, [-1, 1]) = &[10, 8.7, 7.4, 6.3, 5.2, 4.3, 3.4, 2.7, 2, 1.4, \\ &1.0, 0.6, 0.3, 0.1, 0, 0, 0.1, 0.3, 0.6, 1.0, 1.4, 2.0]. \end{aligned} \tag{136}$$



(a) The polynomial p described in the Bernstein basis of degree 2.

(b) The polynomial p described in the Bernstein basis of degree 21.

Figure 16: The polynomial $p(x) = 5x^2 - 4x + 1$ described in the Bernstein basis of degree 2 and 21 on the simplex $\sigma = [-1, 1]$. A green cross indicates a positive coefficient, a red cross indicates a negative coefficient, and a blue cross indicates a coefficient equal to zero. Note how the control points are converging to the graph of p.

The minimum value of the entries is exactly equal to zero, for both $b_{(7,14)}$ and $b_{(6,15)}$. Since neither of these are at a vertex, $b(p, 21, [-1, 1])$ is a (local) certificate of positivity for p. The polynomial in the two descriptions is shown in Figure 16. A green cross indicates a positive coefficient, a red cross indicates a negative coefficient, and a blue cross indicates a coefficient equal to zero. ■

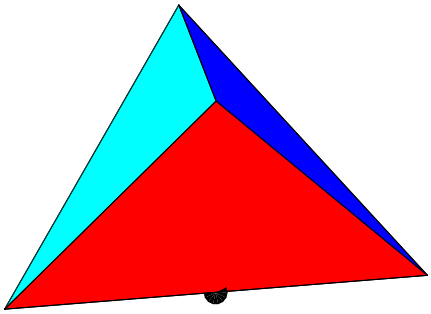
11 By Sub-Division

The process of splitting a simplex into two or more smaller simplices, is called sub-division. This is obtained by placing a new vertex on the original simplex and then considering the original simplex as a collection of simplices. The new vertex can be placed either on a lower dimensional face or in the interior of the original simplex. These variations are shown in Figure 17.

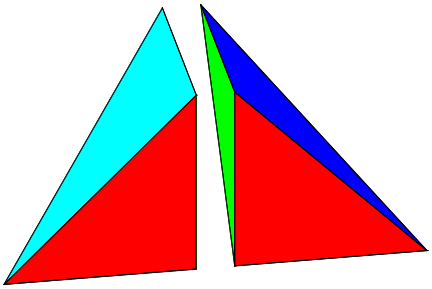
In Figure 17(a), a 3-simplex is shown with a new vertex (indicated by the black ball) placed on the 1-face at the bottom. Figure 17(b) shows the resulting two 3-simplices in an exploded view for clarity.

In Figure 17(c), the 3-simplex is shown with the new vertex (the black ball) placed on the 2-face, coloured in red. Note that since $n = 3$, the 2-face is actually a facet of the simplex. Figure 17(d) shows the resulting three 3-simplices in an exploded view for clarity.

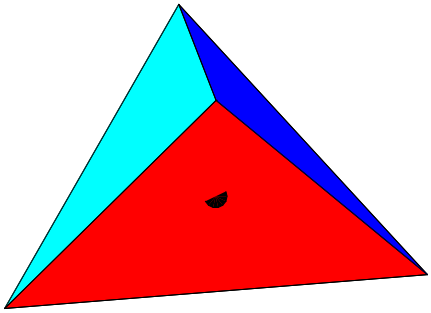
The case where the new vertex is placed in the interior of the original simplex does not allow for a clear presentation, and is thus omitted. In general, sub-dividing by placing the new vertex in the interior of an n-simplex results



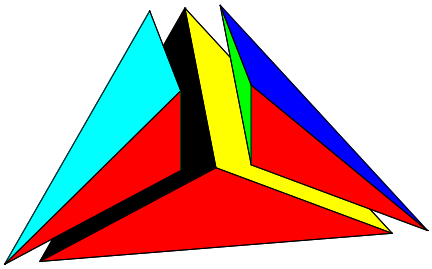
(a) Three dimensional simplex with the new vertex placed on the 1-face at the bottom.



(b) The resulting collection of (two) simplices after sub-divison at the new vertex.



(c) Three dimensional simplex with the new vertex placed on the red facet in the front.



(d) The resulting collection of (three) simplices after sub-divison at the new vertex.

Figure 17: Simplices before and after sub-division. Observe that placing a new vertex on a 1-face results in two simplices, and that placing a new vertex on a 2-face results in three simplices.

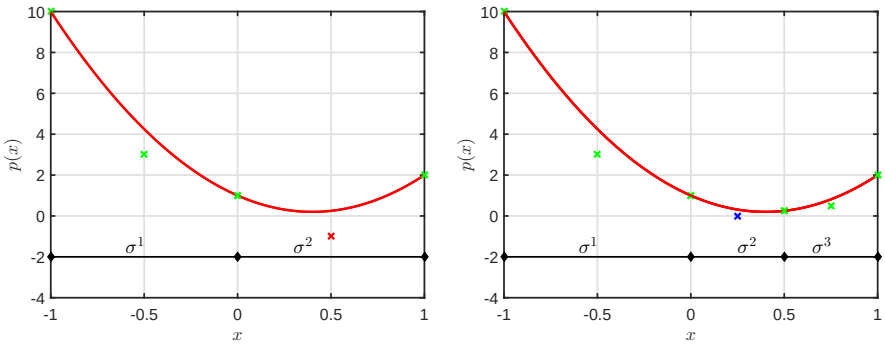
in $n+1$ simplices in the resulting collection.

As was the case for degree elevation, the motivation for investigating positivity of polynomials by sub-division is Bernstein's Theorem.

Theorem 10 *(Bernstein's Theorem: Sub-division [26]) Let the polynomial p, of actual degree d, be positive on the simplex σ. Then there exists a sub-division of σ into a collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ containing finitely many simplices such that $C > 0$, where C is the vector of all coefficients in the collection, e.g.*

$$C^i = b(p, d, \sigma^i) > 0 \qquad \forall \sigma^i \in K. \tag{137}$$

Bernstein's Theorem ensures the ability to certify the positivity of positive polynomials by sub-division, and the (local) certificate of positivity is the coefficient vector C. A theoretical bound on the number of simplices m in the collection is derived in [26]. The bound relies of the minimum of the polynomial on the original simplex, and on the way one chooses to place new



(a) Simplex σ after the first sub-division into the collection of simplices $K = \{\sigma^1, \sigma^2\}$.

(b) Simplex σ after the second sub-divsion into the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3\}$.

Figure 18: Positivity certification of $p(x) = 5x^2 - 4x + 1$ on the simplex $\sigma = [-1, 1]$ using sub-division. The colours used for the control points indicates a positive, negative or zero coefficient.

vertices. This choice is of tremendous interest, as it offers a way of tweaking the certification process. After the following example, different strategies for vertex placement are presented in the next section, which composes most of the remainder of this Chapter.

Example

Revisiting the polynomial $p(x) = 5x^2 - 4x + 1$ from above, it is still wished to investigate the local positivity of p on the simplex $\sigma = [-1, 1]$. The coefficient vector $b(p, 2, [-1, 1]) = [10, -4, 2]$ could not be a certificate of positivity, and to show the (local) positivity of p, sub-division is used. To place new vertices, the chosen strategy is simply to half the line segments. This is in fact the case of one dimensional binary splitting, see below.

Figure 18 shows the process. First, on Figure 18(a), the original simplex σ is sub-divided into collection of simplices $K = \{\sigma^1, \sigma^2\}$. Since $b(p, 2, [-1, 0])$ $= [10, 3, 1]$ it is a certificate of positivity of p on σ^1. Since $b(p, 2, [0, 1])$ has a negative element, σ^2 needs further sub-division. On Figure 18(b) σ^2 has been sub-divided, resulting in the collection of simplices $K = \{\sigma^1, \sigma^2, \sigma^3\}$. The minimum of $b(p, 2, [0, 0.5]) = [1, 0, 0.25]$ is zero at a grid point not located at a vertex, and $b(p, 2, [0.5, 1]) = [0.25, 0.5, 2]$. Thus the three coefficient vectors $b(p, 2, \sigma^i)$ are certificates of positivity of p on their respected simplices. Together, $C = [10, 1, 0.25, 2, 3, 0, 0.5]$ is a (local) certificate of positivity of p on σ.

■

This example shows one of the features of sub-division which makes it favourable compared to degree elevation. Once $b(p, d, \sigma^i)$ certifies the positivity of p on σ^i, σ^i is removed from the remaining certification process. This allows for a concentrated process, compared to degree elevation, where the entire original simplex continues to be investigated until it is certified.

12 Vertex Placement

This section considers the choice of *where* to place new vertices when certifying positivity of polynomials using sub-division. Any such choice will result in a sub-division routine. There are two types of sub-division routines.

The first type of sub-division routine is to determine a pre-defined pattern and apply it on a given simplex, until the certificate of positivity is obtained. Using a pre-defined pattern offers the advantage of ensuring convergence properties of the sub-division. This is, for instance, needed in proving the bound on the number of simplices m in the collection mentioned above.

The other way is adaptive by determining a way of analysing the given polynomial, and figuring out a way to incorporate the information to guide the choice of where to place the next vertex. Analysing the polynomial at hand offers the advantage of being able to obtain a certificate of positivity using fewer sub-divisions than a pre-defined pattern would have to use, but with limited or perhaps even no insurance that the chosen way of incorporating the information will result in a certificate at all.

Shrinking Factor

For sub-division routines using a pre-defined pattern, there is a measure of "how much" smaller the simplices are after one sub-division. This is related to the diameter of simplices in a collection, see Definition 2. With this, the shrinking factor is as follows.

Notation 11 *Let σ be a simplex and K be the collection of simplices after applying a sub-division routine once. Then the shrinking factor $C \in (0, 1]$ of the sub-division routine is given by*

$$Ch(\sigma) = h(K). \tag{138}$$

Thus C is a measure of how much the diameters of the simplices in a collection shrinks by applying the routine once. I will refer to any sub-division routine with a constant C with $0 < C < 1$ as a regular sub-division routine. This is the case for all sub-division routines utilising a pre-defined pattern, and two such routines are the standard triangulation and the binary splitting, see below.

For routines which utilise information from an analysis of the given polynomial, the shrinking factor may vary from one sub-division to the next. Also, the routine may occasionally sub-divide without changing the diameter or it may leave some simplices in the collection undivided, resulting in a shrinking factor of $C = 1$. This is highly dependent on which information is obtained, and how it is used in the routine.

12.1 Pre-Defined Vertex Placement

In the following, two strategies using a pre-defined pattern for sub-division are presented.

Standard Triangulation

Standard triangulation[1] [17] is perhaps the most regular sub-division routine one can imagine. It consists of placing a new vertex in the middle of each 1-face. In general, this results in one simplex being sub-divided into 2^n simplices. Besides the easy determination of new vertices, this approach offers the feature that the volume of the resulting simplices are 2^{-n} times the volume of the original simplex.

On Figure 19, the standard triangulation is performed on the three dimensional standard simplex, see Equation (25). In [26] the shrinking factor of the standard triangulation applied to the standard simplex is shown to be $C \leq \frac{\sqrt{n}}{2\sqrt{2}}$.

Binary Splitting

Binary splitting [26] is even simpler than standard triangulation. It consists of placing one new vertex in the middle of the longest 1-face of the original simplex. This is not necessarily unique, and in the case of two or more longest 1-faces, one of them is chosen arbitrarily. This always results in one simplex being sub-divided into 2 simplices.

Binary splitting of a three dimensional simplex was already shown on Figure 17. In [26] the shrinking factor is show to be $C = \frac{1}{2}$ after $\frac{n(n+1)}{2}$ applications of the binary splitting. Since the shrinking factor after one application of the binary splitting depends on the second longest 1-face the shrinking factor is in fact not constant. However, considering the use of the binary splitting $\frac{n(n+1)}{2}$ times as one application, the shrinking factor has a constant upper bound. This enabled the use in the convergence analysis.

[1] In this Thesis, triangulation is the splitting of a box into simplices, but earlier works considered the word triangulation for both that, and for what I call sub-division. Should the naming be aligned to my conventions, standard triangulation should be named standard sub-division. However, the name is kept for historical reasons.

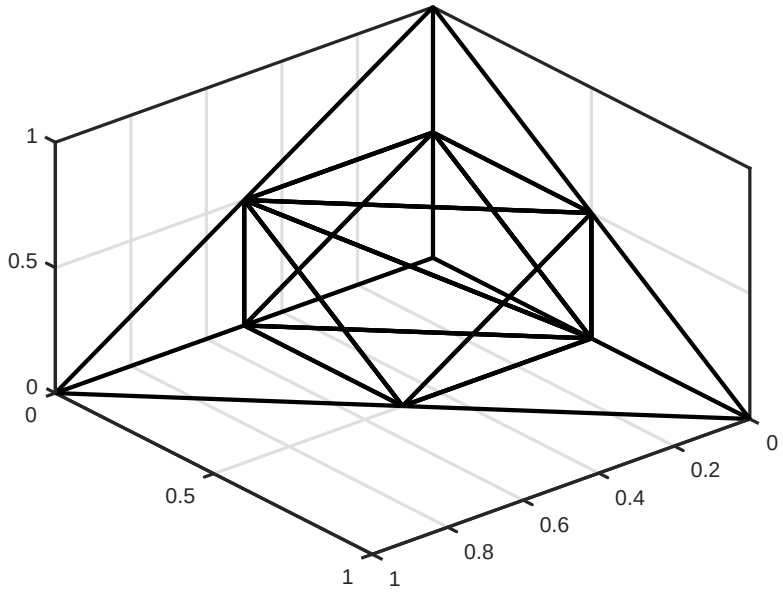


Figure 19: Standard triangulation of the standard simplex in three dimensions.

12.2 Adaptive Vertex Placement

At the offset of my PhD studies, the insight about how to obtain information from a given polynomial was limited. I set out to investigate this, and to familiarise myself with the mechanisms governing different strategies for vertex placement. On a somewhat philosophical level, it was always my belief that there had to be some "optimal" way of utilising the given polynomial, that the polynomial wanted to be certified positive. I worked with "optimal" meaning either having a fast run-time, which is implementation dependent, or needing the fewest simplices in the collection, which is implementation independent.

The sub-division routines presented in the following, are most of the strategies I considered for vertex placement in the beginning of my studies. Most of them are variations of each other, with different advantages and drawbacks.

Grid Point Triangulation

The strategy of grid point triangulation is thanks to Christoffer Sloth. He presented it to me at the beginning of my studies and he had been using it without considering its possible improvement. The strategy is to place the next vertex at the grid point with the most negative coefficient. The strategy is derived based on a simple heuristic. It is fair to assume that the minimum of

the polynomial is "close" to the grid point with the most negative coefficient. Then, by the end-point value property, placing a vertex at the grid point of the most negative coefficient offers an easy way of obtaining an evaluation of the polynomial at that point. If the coefficient at the vertex in the description on the new collection is negative, then the certification process can stop and return an explicit point at which the polynomial is in fact negative. This offers a natural stopping criterion and is another advantage compared to certifying using degree elevation.

Scaled Grid Point Triangulation

Since the basis polynomials are unimodal (see Chapter Bernstein Basis Properties), I investigated a possible improvement of the grid point triangulation. The maximal value of the basis polynomials differ according to their α-index. By scaling the entries of the coefficient vector accordingly, the coefficient which, together with the basis polynomial, contributes most negatively to the graph of the polynomial, can be identified. This is shown in the next example, which also exemplifies grid point triangulation.

Example

On Figure 20, the polynomial

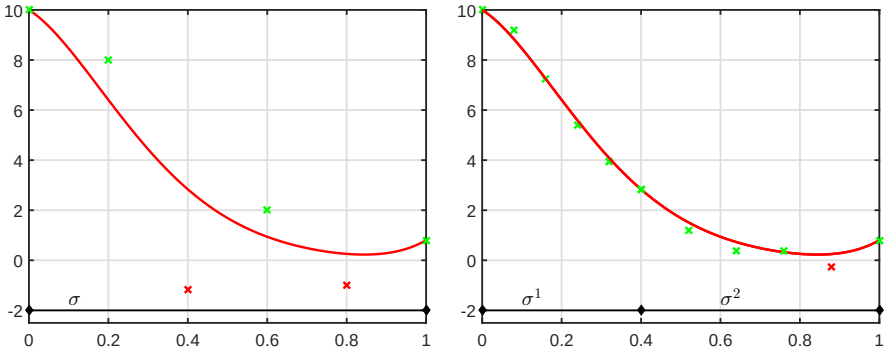
$$p(x) = 67.6x^5 - 190.4x^4 + 195.4x^3 - 71.8x^2 - 10x + 10 \tag{139}$$

is shown. It is wished to investigate p for local positivity on the simplex $\sigma = [0, 1]$. When described in the Bernstein basis of degree $D = d = 5$ the coefficient vector is $b(p, 5, [0, 1]) = [10, 8, -1.18, 2, -1, 0.8]$. The most negative coefficient is $b_{(3,2)} = -1.18$. Using grid point triangulation will thus put the new vertex at $\Delta_{(3,2)} = 0.4$. This is shown in Figure 20(b). As seen, one application of the grid point triangulation does not certify the positivity of p on σ as $b(p, 5, [0.4, 1])$ still contains a negative entry, $b_{(1,4)}(p, 5, [0.4, 1]) = -0.28$. Applying grid point triangulation on σ^2 at $\Delta_{(1,4)} = 0.88$ yields Figure 20(d) where p is certified positive on σ.

Instead, if scaled grid point triangulation is used, the negative elements of $b(p, 5, [0, 1])$ are scaled according to the corresponding basis polynomial. By the unimodality property, the maximal values of the basis polynomials are obtained at the corresponding grid points. Thus

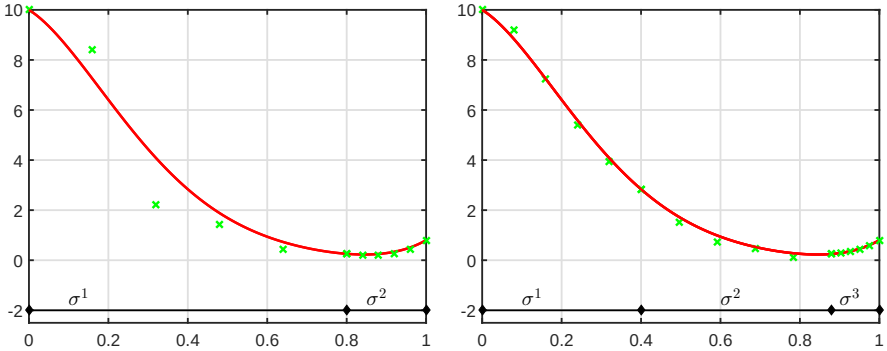
$$\mathcal{B}^5_{(3,2)}([0, 1])|_{\Delta_{(3,2)}(5,[0,1])} = 0.3456, \tag{140}$$

$$\mathcal{B}^5_{(1,4)}([0, 1])|_{\Delta_{(1,4)}(5,[0,1])} = 0.4096. \tag{141}$$



(a) Initial description of p on σ.

(b) Description after one application of grid point trinagulation.



(c) To certify the positivity of p only one application of the scaled grid point triangulation is needed.

(d) Description after another application of grid point trinagulation of σ^2 in Figure (b).

Figure 20: The polynomial p under two strategies for positivify certification. Figure (b) shows the result after one application of the grid point triangulation. Directly below it, Figure (d) shows the result after another application. Figure (c) shows the result of the scaled grid point triangulation.

Since

$$b_{(3,2)}(p,5,[0,1])\mathcal{B}^5_{(3,2)}([0,1])|_{\Delta_{(3,2)}(5,[0,1])} > b_{(1,4)}(p,5,[0,1])\mathcal{B}^5_{(1,4)}([0,1])|_{\Delta_{(1,4)}(5,[0,1])}, \tag{142}$$

the scaled grid point triangulation will select the new vertex to be placed at $\Delta_{(1,4)} = 0.8$. This is shown in Figure 20(c). As seen, the scaled grid point triangulation certifies the positivity of p using one sub-division, resulting in two simplices. Since grid point triangulation used three simplices, the scaled grid point triangulation performed better in this example.

■

Experience from working with the scaled grid point triangulation revealed that the strategy only rarely altered which grid point to place a new vertex at, compared to grid point triangulation. When it did do so, the number of needed simplices before a certificate was obtained was not necessarily improved.

Extremums Placement

An inherent feature of (scaled) grid point triangulation is the fact that the new vertex can only be placed at a finite set of points defined by the grid points. In general there are

$$N_D - (n+1) \tag{143}$$

possible points, where $N_D = \binom{n+D}{n}$ is the number of grid points and $n+1$ is the number of vertices. In an attempt to overcome this and be able to place the new vertex more freely, I developed the following strategy, coined extremums placement.

Presented with a coefficient vector with more than one negative entry, define the polynomial

$$p_{\text{neg}} = \sum_{\alpha \in \mathcal{I}} b_\alpha(p, D, \sigma) \mathcal{B}_\alpha^D(\sigma) \tag{144}$$

where $\mathcal{I} = \{\alpha | b_\alpha(p, D, \sigma) < 0\}$. Thus p_{neg} is composed of all the negative contributions to p. Then find the minimum of p_{neg} on σ and use the corresponding point as a new vertex.

Although extremums placement successfully improved the placement of the new vertex and often resulted in a decrease in the number of needed simplices in small examples, the strategy suffers from another issue. The task of certifying positivity of one polynomial is transformed into determining the minimum of another polynomial. Regardless of which technique is chosen to determine the minimum of p_{neg}, one could just as well determine the minimum of p directly and obtain the wanted certificate of positivity in that way. Note that p_{neg} is of the same degree and dimension as p. Thus the task of determining the minimum of them are of similar difficulty.

Newton Guess Correction

Inspired from optimisation, it was attempted to consider the grid point of the most negative coefficient as a hot start in a minimisation. It was tried with both the polynomial and its derivative as objective function, and different search direction and step size determination methods were considered. No combination of them were particularly obvious based on the mathematics involved, and none of them really stood out as particularly useful. In addition,

they all require symbolic manipulation of high degree and high dimensional polynomials, causing the strategy to run slow, if at all.

The Newton guess correction is named after classical and modified Newton's methods from optimisation, see e.g. [5]. For the reasons above, the Newton guess correction was abandoned, but the work was not wasted. It brought to my attention that the negativity of some coefficients contains more information than the negativity of other coefficients does. This resulted in the next strategy.

Information Exhaustion

This strategy relies on the fact that a polynomial restricted to a face is defined by the coefficients on that face. This was seen in the example in Section 2.1. This results in the following observation: A polynomial with negative coefficients on one of its faces will continue to have negative coefficients on that face, until that face is sub-divided. Hence, a sub-division which does not sub-divide all faces containing at least one negative coefficient can never result in a certificate of positivity, and there will always be a need for additional sub-division. For this reason the most negative coefficient is not necessarily the most interesting coefficient.

Where the grid point triangulation would eventually sub-divide all faces containing negative coefficients, this strategy simply sub-divides all faces containing a negative coefficient right away. If a face has more than one negative coefficient, the most negative of those coefficients is used. Thus this strategy exhausts all the information given in one coefficient vector, hence the name.

For simplices of dimension 3 or higher, an interesting effect occurs for this strategy. If there are negative coefficients on faces of different dimensions, the order in which the faces are sub-divided starts to play a role. To see this, consider Figure 17. On Figure 17(b), the sub-division is performed on a face of dimension 1, but the sub-division also divides the red face of dimension 2. On Figure 17(d), the sub-division is performed on the red face of dimension 2, but this time the sub-division does not divide the face of dimension 1 at the bottom. This calls for a choice of whether to sub-divide low-dimensional faces first, or to place vertices at the grid points with the most negative coefficient, and then proceed to sub-divide until all faces (of the original simplex) containing a negative coefficient have been sub-divided.

Some initial examples did not reveal a clear preference regarding this choice, but the strategy of information exhaustion was a great improvement compared to grid point triangulation, both on run-times and on the number of needed simplices to obtain the certificate of positivity. In the implementation which followed, the grid point with the most negative coefficient is used to place the first new vertex. Then, the grid point with the second most

negative coefficient is used to place the next new vertex, given that the face of this grid point has not already been sub-divided. The implementation then proceeds through all negative coefficients until all faces of interest have been sub-divided.

Equal to Zero Approach

Although information exhaustion was chosen as the strategy for positivity certification in the rest of my work, one more strategy deserves to be mentioned in this context. Where the strategies above all abide to some level of "let us see what happens" by (intelligently) placing one or more new vertices and then analysing the coefficient vector of the resulting collection, this strategy flips the certification process entirely.

In one dimension, the equal to zero approach aims at designing a collection of simplices where the simplices are as big as they can be. This is done by extracting the equation of the most negative coefficient, and then allowing for *one* vertex to be a variable in the equation. Equating to zero then reveals where to place the vertex such that the most negative coefficient (in the original simplex) becomes equal to zero (in the new simplex). In one dimension, this strategy naturally outperforms all other strategies when comparing by the number of needed simplices. However, when the dimension is two or higher, *one* equation to equate to zero no longer suffices to determine the place to put a new vertex. The vertex used as a variable now has two or more components, one for each dimension, and additional equations are needed to obtain a solvable system of equations. Despite my best efforts, have I not been able to identify any equations suitable to govern the vertex placement which scale with dimension.

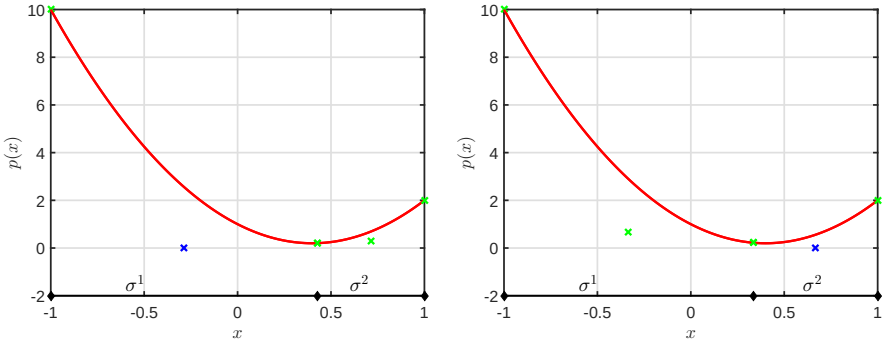
Example

Consider the polynomial $p(x) = 5x^2 - 4x + 1$ which was investigated for positivity above, using both degree elevation and as the introductory example to sub-division. The simplex of interest was $\sigma = [-1, 1]$ and the initial coefficient vector was $b(p, 2, [-1, 1]) = [10, -4, 2]$. In Appendix B, equations for basis transformation between monomial and Bernstein bases are derived. Using Equation (B.17) reduced to one dimension yields

$$b_{(1,1)}(p, 2, [\sigma_0, \sigma_1]) = 5\sigma_0\sigma_1 - 4\left(\sigma_0\frac{1}{2} + \sigma_1\frac{1}{2}\right) + 1. \tag{145}$$

Choosing the variable as σ_1 from the original simplex σ gives

$$b_{(1,1)}(p, 2, [-1, \sigma_1]) = 0 \Rightarrow \sigma_1 = \frac{3}{7}, \tag{146}$$



(a) Sub-division obtained by treating σ_1 as a variable. Maximises the resulting σ^1.

(b) Sub-division obtained by treating σ_0 as a variable. Maximises the resulting σ^2.

Figure 21: Collection of simplices obtained from the equal to zero approach. Either vertex σ_1 or vertex σ_0 in the original simplex is treated as a variable.

such that

$$b(p, 2, [-1, \tfrac{3}{7}]) = [10, 0, 0.2041] \tag{147}$$
$$b(p, 2, [\tfrac{3}{7}, 1]) = [0.2041, 0.2857, 2]. \tag{148}$$

Choosing the variable as σ_0 from the original simplex σ gives

$$b_{(1,1)}(p, 2, [\sigma_0, 1]) = 0 \Rightarrow \sigma_0 = \frac{1}{3}, \tag{149}$$

such that

$$b(p, 2, [-1, \tfrac{1}{3}]) = [10, 0.667, 0.2222] \tag{150}$$
$$b(p, 2, [\tfrac{1}{3}, 1]) = [0.2222, 0, 2]. \tag{151}$$

As seen, the strategy does indeed maximise the resulting simplex. In fact, choosing any point in the interval $[\frac{1}{3}, \frac{3}{7}]$ as the new vertex will result in a sub-division of σ into two simplices where neither description contains negative coefficients. It is interesting to observe that the derivative changes sign at $x = \frac{2}{5}$, which is inside the interval as well. The situation is shown in Figure 21.

■

13 By Dimension Elevation

During my stay in Colorado in 2016, my host Sriram Sankaranarayanan came up with the idea that there perhaps was a third way of obtaining positivity

certificates. By artificially expanding the dimension of a given polynomial, the resulting coefficient vector would contain more entries corresponding to the extra control points in the control net. I agreed that it would be of interest to investigate whether there was any information to be obtained from the extra coefficients.

After a few iterations, including presenting the idea to Rafał and Christoffer upon my return to Aalborg, the final result revealed itself. The information obtained by elevating the degree is in some sense equivalent to the information obtained from sub-division. This equivalence is derived in Appendix C, and dimension elevation is not considered any further.

14 Gap Between Positive Definite and Bernstein Basis Certifiable

In the coming stability analysis, it is of interest to certify non-negativity and positive definiteness, but the Bernstein Theorem does not encompass this situation completely. Polynomials can be positive definite without any degree or sub-division to show it. An explicit example of this is treated in Appendix D. This is an inherent drawback of using the Bernstein basis in stability analysis. However, as shall be evident in Part II, the Bernstein basis still offers some highly desired advantages compared to the monomial basis. The most obvious advantage is that the implication of the Bernstein Theorem holds, i.e. a polynomial defined on a collection of simplices with one vertex coefficient equal to zero, the other vertex coefficients positive, and the remaining coefficients non-negative, will be positive definite. To distinguish, I introduce the following notion.

Notation 12 *(Bernstein Basis Certifiable) Let $p : U \to \mathbb{R}$, with $U \subset \mathbb{R}^n$ a closed polytope, be a positive definite polynomial ($p \in \mathbb{R}_{>0}[x]$) with one point $x^* \in U$ such that $p(x^*) = 0$. If there exists a collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ covering U such that $C^i \geq 0$ for all $i \in \{1, 2, \ldots, m\}$, when p is described on K, then p is said to be Bernstein basis certifiable. Such a collection is said to be a Bernstein basis certifying collection for p.*

As a natural consequence, the point x^* needs to be at a vertex in K.

For completeness, I mention that there is a difference between a polynomial being positive definite and a sum of squares, in the sense that a polynomial can be positive definite without being able to be expressed as a sum of squares. This difference was first proved in 1888 by Hilbert, but not until 1967 did an explicit example occur in the literature, thanks to Motzkin [43]. However, on compact domains, positive definite polynomials can be approximated to arbitrary precision by sum-of-squares polynomials of higher

degree. This is due to the density of sum-of-square polynomials, a result given by Lasserre in [25].

15 Software

The function *getPositivityCertificate* takes five inputs defining a polynomial in the monomial basis and returns a certificate of positivity. C, $gamma$, vex, and $simplex$ define the polynomial and the initial collection of simplices on which the positivity is wished to be investigated. Due to numerical issues, is it sometimes of interest to specify a tolerance on the coefficient evaluation. This can be done with the *tol* argument. If left unspecified, the default is zero tolerance. The function then proceeds to investigate the positivity of the polynomial on each simplex in the collection. If the certificate cannot be obtained on the current simplex, it is sub-divided according to the grid point triangulation and the information exhaustion strategies. The function continues sub-dividing until a certificate is obtained, or an explicit point at which the polynomial is negative is identified. The output CB is the coefficient vector of the polynomial when described on the resulting (Bernstein basis certifying) collection, defined by vex and $simplex$. The α-matrix, the degree, and the dimension are $alpha$, d, and n respectively.

The function *GPT_ExhaustingInfo* takes eight inputs defining a collection of simplices, and returns a collection of simplices where one simplex has been sub-divided according to the grid point triangulation and information exhaustion strategies. The inputs are: $alpha$ is the α-matrix, I is the index of the most negative coefficient, d is the degree, n is the dimension, vex and $simplex$ define the collection of simplices, i is the number of the simplex in the collection which is to be sub-divided, and CB is the coefficient vector of the polynomial on the i^{th} simplex. The outputs are vex and $simplex$ which define the new collection of simplices.

Part II

Stability Analysis

Stability Introduction

The second part of this Thesis is concerned with dynamical systems, in particular their equilibrium points and the (in)stability of them. The notion of stability offers several interpretations and, depending on the setting, they all have their merit. This Thesis solely focuses on continuous time polynomial dynamical systems and stability in the sense of Lyapunov. This will be specified later in this Chapter, but first some definitions are needed.

16 Definitions

Besides the definitions from Part I, the following well-known definitions are needed in this Part.

16.1 Comparison Function

For later definition and classification of different stability notions, the concept of comparison functions greatly simplify the notation. A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{K}$ if it is continuous, zero at zero, and strictly increasing. A function $\sigma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{L}$ if it is continuous, strictly decreasing, and $\lim_{t\to\infty} \sigma(t) = 0$. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to be of class $\mathcal{KL}$ if it is class $\mathcal{K}$ in its first argument and class $\mathcal{L}$ in its second argument.

Combined with any p-norm, class $\mathcal{K}$ comparison functions can be extended to n variables in what is termed a monotone aggregation function $\mu : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$, such that $\mu(x) = \alpha(||x||)$. Similarly, class $\mathcal{KL}$ comparison functions can be extended to n variables in the first argument to $\theta : \mathbb{R}^n_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, such that $\theta(x,t) = \beta(||x||,t)$. Throughout the Thesis, I shall solely use the $\alpha(||x||)$ and $\beta(||x||,t)$ constructions, and reference to both as comparison functions. See [22] for a detailed exposition of comparison functions.

16.2 Vector Field

Let $f : U \to \mathbb{R}^n$ with $U \subseteq \mathbb{R}^n$. Then, for

$$\Sigma : \dot{x} = f(x), \tag{152}$$

Σ is termed a dynamical system and f is termed the vector field. The vector of variables x will commonly be referred to as the state or the state variables and U is the state space. Given a set of initial conditions x_0 at some initial time t_0, the solution to the differential equation (152) is noted $x(t)$. The path followed in state space of a solution $x(t)$ for $t \geq t_0$ is called the solution trajectory, or simply trajectory. Henceforth, the initial time will be assumed to be $t_0 = 0$.

As I only consider polynomial vector fields, $f \in \mathbb{R}[x]$ throughout, where $\mathbb{R}[x]$ is the ring of polynomials. The familiar reader will recognise Σ as an unforced system, or a system consisting solely of a drift term. In this Thesis, only the uncontrolled case is considered, or at least the case of a known controller where the system has already been simplified to a pure drift term.

16.3 Equilibrium Point

An equilibrium point is a state x^* satisfying

$$f(x^*) = 0 \tag{153}$$

meaning that the vector field vanishes at x^*. This also indicates that the dynamics of Σ does not evolve, should the equilibrium point ever be reached.

Without loss of generality, if nothing else is mentioned, it will be assumed that all equilibrium points are located at the origin, $x^* = 0$. By a linear change of variables, equilibrium points can be translated to an arbitrary location without changing the stability properties of the equilibrium, see e.g. [30] for an example of this.

17 Lyapunov Stability

Remembering the ball of radius r being B_r, the notion of stability in the sense of Lyapunov can be expressed as follows.

Definition 13 *(Stability in the Sense of Lyapunov [47]) The equilibrium point $x = 0$ is said to be stable if, for any $R > 0$, there exists $r > 0$, such that if $x_0 \in B_r$, then $x(t) \in B_R$ for all $t \geq 0$. Otherwise, the equilibrium point is unstable.*

In words, the definition reads that if the initial state is sufficiently close to the origin in terms of Euclidean distance, the solution trajectories will stay within a maximal Euclidean distance from the origin for all future time. But

staying close to the origin is generally not a sufficiently satisfying condition; convergence (of some sort) of the state variables to the equilibrium is required.

Definition 14 *(Asymptotic Stability [47]) A stable equilibrium point is said to be asymptotically stable if the trajectories for $x_0 \in B_r$ also implies that $x(t) \to 0$ for $t \to \infty$.*

Asymptotic stability is arguably the weakest form of stability, since the convergence rate can be infinitely slow. Later, this Chapter presents two other types of stability with faster convergence rates, but first an essential tool for determining (asymptotic) stability is presented.

17.1 Lyapunov Function

If a function $V : U \to \mathbb{R}$ is positive definite with continuous partial derivatives and has a negative semi-definite time derivative $\dot{V}(x)$ along the solution trajectories of Equation (152), then $V(x)$ is called a Lyapunov function. By the chain rule

$$\dot{V}(x) = \frac{dV(x)}{dt} = \frac{\partial V(x)}{\partial x}\dot{x} = \frac{\partial V(x)}{\partial x}f(x) \tag{154}$$

which gives rise to the term derivative along trajectories. In fact, the Lie derivative of $V(x)$ with respect to $f(x)$ is $L_V(f) = \frac{\partial V(x)}{\partial x}f(x)$ which is why $\dot{V}(x)$ will often also be referred to simply as the Lie derivative.

Theorem 15 *(Lyapunov Function for (Asymptotic) Stability [47]) Let the origin be an equilibrium point of the dynamical system Σ. If there exists a Lyapunov function $V(x)$ the equilibrium is stable. Asymptotic stability is obtained if the Lie derivative is negative definite.*

An equivalent theorem for asymptotic stability can be given using comparison functions as follows.

Theorem 16 *(Asymptotic Stability using Comparison Functions [23]) Let the origin be an equilibrium point of the dynamical system Σ. If there exists a function $V : U \to \mathbb{R}$ such that*

$$\begin{aligned} \alpha_1(||x||) \leq V(x) \leq \alpha_2(||x||) \\ \dot{V}(x) \leq -\alpha_3(||x||), \end{aligned} \tag{155}$$

where $\alpha_i(||x||)$ are class $\mathcal{K}$ comparison functions, then the origin is asymptotically stable.

The decay of the trajectories to the origin is evident from the inequality

$$||x(t)||_2 \leq \beta(||x_0||, t) \tag{156}$$

which can be obtained by manipulating (155) and using the comparison lemma for comparison functions ([23]). Here $\beta(||x_0||, t)$ is a class $\mathcal{KL}$ function. The equivalence between the Lyapunov function in Theorem 15 and the function V in Theorem 16 should be evident.

This result is part of Alexandr Lyapunov's original work from 1892. Later on, the work on converse Lyapunov theory affirmatively answered the question on the necessity of the existence of a Lyapunov function for an equilibrium to be asymptotically stable. See e.g. [18] or [34].

In modern notation, and fitted to the use in this Thesis, a converse theorem taken from [6] reads as follows.

Theorem 17 *(Converse Lyapunov Result for Asymptotic Stability [6]) Let the origin be an equilibrium point of the dynamical system Σ, and let it be asymptotically stable. Then there exists a C^∞ Lyapunov function with a negative definite Lie derivative.*

Here C^∞ is the class of infinitely differentiable functions.

With the existence of a Lyapunov function established as a necessary and sufficient condition for asymptotic stability, the next natural question is *how* to obtain such a function for a given system. The vigilant reader will by now have realised that this Thesis is a humble attempt at answering that question.

17.2 Lyapunov Functions in the Bernstein Basis

This section introduces the link between the properties of the Bernstein basis and Lyapunov functions. Showing positive definiteness of general non-linear multivariate functions is a difficult problem. However, besides only considering polynomial dynamical systems, restricting the Lyapunov function to be polynomial, $V(x) \in \mathbb{R}[x]$, allows for an easy use of the Bernstein basis properties to show positive definiteness of $V(x)$ and negative definiteness of $\dot{V}(x)$. How to do this was covered in Chapter Positivity Certification.

Before I initiated my studies, Rafał and Christoffer had already translated the Lyapunov criteria from above to conditions on the Bernstein coefficients. With the notation introduced in Part I, the result reads as follows.

Lemma 18 *([46]) Let $V(x)$ be a polynomial of degree d_V defined on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ according to Definition 6 as*

$$V(x) = CV^i \mathcal{B}^{d_V}(\sigma^i) \qquad \forall x \in \sigma^i, i \in \{1, 2, \ldots, m\}, \tag{157}$$

$$CV^{i^T} = {}^iB_j \, CV^{j^T} \qquad \forall\, i, j \in \{1, \ldots, m\}, \tag{158}$$

where CV is the row vector of all coefficients in the collection and CV^i denotes the coefficients on the i^{th} simplex. For some dynamical system Equation (152)*, also described on the collection K, let $\dot{V}(x)$ be the Lie derivative. It is a polynomial of degree d_L defined on K according to Definition 6 as*

$$\dot{V}(x) = CL^i \mathcal{B}^{d_L}(\sigma^i) \qquad \forall x \in \sigma^i, i \in \{1, 2, \dots, m\}, \tag{159}$$

$$CL^{i^T} = {}^iB_j \, CL^{j^T} \qquad \forall\, i, j \in \{1, \dots, m\}. \tag{160}$$

Assume, without loss of generality, that the simplices are numbered such that the origin is a vertex of the first $\bar{m}$ simplices. If

$$CV \geq 0 \tag{161a}$$

$$CV^i_{d_V e_0} = 0 \quad \forall i \in \{1, \dots, \bar{m}\} \tag{161b}$$

$$CV^i_{d_V e_j} > 0 \quad \forall i \in \{1, \dots, \bar{m}\}, \forall j \in \{1, \dots, n\} \tag{161c}$$

$$CV^i_{d_V e_j} > 0 \quad \forall i \in \{\bar{m}+1, \dots, m\}, \forall j \in \{0, \dots, n\} \tag{161d}$$

$$CL \leq 0 \tag{161e}$$

$$CL^i_{d_L e_0} = 0 \quad \forall i \in \{1, \dots, \bar{m}\} \tag{161f}$$

$$CL^i_{d_L e_j} < 0 \quad \forall i \in \{1, \dots, \bar{m}\}, \forall j \in \{1, \dots, n\} \tag{161g}$$

$$CL^i_{d_L e_j} < 0 \quad \forall i \in \{\bar{m}+1, \dots, m\}, \forall j \in \{0, \dots, n\}, \tag{161h}$$

then $x = 0$ is a local asymptotically stable equilibrium point.

Note the necessity of the dynamical system Equation (152) to be described on the collection K. This is always possible, since I only consider polynomial vector fields. This is also why the methods presented in this Thesis cannot be extended to more general non-linear vector field.

As with the positivity certification, which is based solely on the sign of the coefficients, the positive definiteness of the Lyapunov function is embedded completely in its coefficient vector by the convex hull property and the end-point value property. By Equations (161c) and (161d) coefficients at vertices not placed at the origin are positive. By Equation (161b) the coefficient at the vertex at the origin is equal to zero and by Equation (161a) the remaining coefficients are non-negative. The negative definiteness of the Lie derivative is ensured in a similar manner.

17.3 Continuous Piecewise Lyapunov Functions

In the previous sections, Lyapunov functions were considered as functions with continuous partial derivatives. This classification is unnecessarily strict as both Theorem 15 and Theorem 17 can be proven under weaker conditions.

Different manuscripts develop different versions depending on their focus. As the description of polynomials in the Bernstein basis naturally allows for continuous piecewise-polynomials, it was interesting to investigate how to use them as Lyapunov functions. This requires Lyapunov functions which are continuous, but not everywhere differentiable in the usual sense. This situation has been covered in e.g. [18] or [24]. For a modern and thorough treatment developed for the use of continuous piecewise-affine Lyapunov functions see [33].

Allowing for continuous piecewise-polynomials, the limit for the usual derivative does not exist on the facets. Instead, using the formulation of the Dini derivative (see [48]) the use of the supremum (infimum) ensures that the limit always exists. With the definition of a Lyapunov function expanded from "with continuous partial derivatives" to "locally Lipschitz continuous", Theorem 15 still apply. In this case, asymptotic stability is obtained if the Dini derivative is negative definite [33].

When combined with polynomials described in the Bernstein basis, the negative definiteness of the Dini derivative simply amounts to all coefficients being negative. By the Lie derivative, I will refer to both the usual derivative and the Dini derivative in the rest of the thesis. This allows for a natural extension of Lemma 18 to continuous piecewise-polynomial Lyapunov functions as follows.

Lemma 19 *([46]) Let $V(x)$ be a continuous piecewise-polynomial of degree d_V defined on the collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ according to Definition 7 as*

$$V(x) = CV^i \mathcal{B}^{d_V}(\sigma^i) \qquad \forall x \in \sigma^i, i \in \{1, 2, \ldots, m\}, \tag{162}$$

$$\partial_k CV^i = \partial_l CV^j \qquad \forall \left\{ (k, i, l, j) | \partial_k \sigma^i = \partial_l \sigma^j \right\}. \tag{163}$$

For some dynamical system Equation (152)*, also described on the collection K, let $\dot{V}(x)$ be the Lie derivative. It is a discontinuous piecewise-polynomial of degree d_L defined on K according to Definition 8 as*

$$\dot{V}(x) = \begin{cases} \dot{V}_1(x) = b(\dot{V}_1, d_L, \sigma^1)\mathcal{B}^{d_L}(\sigma^1) & \forall x \in \sigma^1 \\ \dot{V}_2(x) = b(\dot{V}_2, d_L, \sigma^2)\mathcal{B}^{d_L}(\sigma^2) & \forall x \in \sigma^2 \\ \cdots & \\ \dot{V}_m(x) = b(\dot{V}_m, d_L, \sigma^m)\mathcal{B}^{d_L}(\sigma^m) & \forall x \in \sigma^m \end{cases}. \tag{164}$$

Assume, without loss of generality, that the simplices are numbered such that the

origin is a vertex of the first $\bar{m}$ simplices. If

$$CV \geq 0 \tag{165a}$$
$$CV^i_{d_V e_0} = 0 \quad \forall i \in \{1, \ldots, \bar{m}\} \tag{165b}$$
$$CV^i_{d_V e_j} > 0 \quad \forall i \in \{1, \ldots, \bar{m}\}, \forall j \in \{1, \ldots, n\} \tag{165c}$$
$$CV^i_{d_V e_j} > 0 \quad \forall i \in \{\bar{m}+1, \ldots, m\}, \forall j \in \{0, \ldots, n\} \tag{165d}$$
$$b(\dot{V}_i, d_L, \sigma^i) \leq 0 \quad \forall i \in \{1, \ldots, m\} \tag{165e}$$
$$b_{d_L e_0}(\dot{V}_i, d_L, \sigma^i) = 0 \quad \forall i \in \{1, \ldots, \bar{m}\} \tag{165f}$$
$$b_{d_L e_j}(\dot{V}_i, d_L, \sigma^i) < 0 \quad \forall i \in \{1, \ldots, \bar{m}\}, \forall j \in \{1, \ldots, n\} \tag{165g}$$
$$b_{d_L e_j}(\dot{V}_i, d_L, \sigma^i) < 0 \quad \forall i \in \{\bar{m}+1, \ldots, m\}, \forall j \in \{0, \ldots, n\}, \tag{165h}$$

then $x = 0$ is a local asymptotically stable equilibrium point.

In Chapter Stability Certification, Lemma 18 and Lemma 19 are used to synthesise Lyapunov functions. At that point, it will become evident why having the option of allowing for a larger class of Lyapunov functions (continuous piecewise-polynomial versus polynomial) is interesting. Before that, the last section of this Chapter covers what is known about the existence of Lyapunov functions with certain specific structures.

18 Existence of Structured Lyapunov Functions

Theorem 17 tells us that for asymptotic stability, the existence of a C^∞ Lyapunov function is a necessary and sufficient condition. However, it does not tell us whether or not we can expect that a polynomial Lyapunov function exists. As it turns out, to sufficiently answer that question, the question needs specification.

So far, I have deliberately avoided specifying the difference between local and global stability. They are quite intuitive. Global asymptotic stability of an equilibrium point is obtained if the initial condition x_0 can be arbitrary and still converge to the origin. In definitions 13 and 14, this amounts to $r > 0$ being arbitrary. Local asymptotic stability of an equilibrium point is obtained if the initial condition x_0 cannot be arbitrary. In this case, convergence to the origin only happens if $x_0 \in B_r$, for some $r > 0$ depending on the system. In accordance with [47] B_r is referred to as *a* domain of attraction, while the largest set of initial conditions which converge is referred to as *the* domain of attraction. Since the boundary of the domain of attraction of a given system in general will be an algebraic curve, the description of polynomials in the Bernstein basis is not suited to analyse domains of attraction. This stems from the barycentric coordinates being defined as affine polynomials.

Another specification is the type of stability under investigation. So far, only asymptotic stability has been presented. Specific choices of comparison functions in Theorem 16 results in more restrictive stability types in which the trajectories decay to the origin with quantifiable rates. One such choice gives exponential stability.

18.1 Exponential Stability

Definition 20 *(Exponential Stability [47]) A stable equilibrium point is said to be exponentially stable if the trajectories for $x_0 \in B_r$ implies that*

$$||x(t)||_2 \leq k||x_0||_2 e^{-\lambda t}, \tag{166}$$

for some constants $k, \lambda > 0$.

As for asymptotic stability, exponential stability can be asserted by the existence of a Lyapunov function.

Theorem 21 *(Exponential Stability using Comparison Functions [23]) Let the origin be an equilibrium point of the dynamical system Σ. If there exists a function $V : U \to \mathbb{R}$ such that*

$$\begin{aligned} \alpha||x||_2^c \leq V(x) \leq \beta||x||_2^c \\ \dot{V}(x) \leq -\gamma||x||_2^c, \end{aligned} \tag{167}$$

with α, β, γ, and c constants > 0, then the origin is exponentially stable.

The exponential decay rate can be shown by manipulating (167) using the comparison lemma for differential inequalities ([23]), and results in

$$||x(t)||_2 \leq \sqrt[c]{\frac{\beta}{\alpha}}||x_0||_2 e^{-\frac{\gamma}{\beta c}t}, \tag{168}$$

which obviously fulfils Equation (166)

Lastly, a converse theorem for exponential stability reads as follows.

Theorem 22 *(Converse Lyapunov Result for Exponential Stability [47]) Let the origin be an equilibrium point of the dynamical system Σ, and let it be exponentially stable. Then there exists a Lyapunov function fulfilling Equation* (167)*.*

Another choice of comparison functions yields rational stability.

18.2 Rational Stability

Definition 23 *(Rational Stability [6]) A stable equilibrium point is said to be rationally stable if the trajectories for $x_0 \in B_r$ implies that*

$$||x(t)||_2 \leq M(1 + ||x_0||_2^k t)^{-1/k}||x_0||_2^\eta, \tag{169}$$

for some constants $M, k > 0$ and $\eta \in (0, 1]$.

In [6] the characterisation of Lyapunov function and the converse result are given in a single theorem as follows.

Theorem 24 *(Converse Lyapunov Result for Rational Stability [6]) The origin of the dynamical system Σ is rationally stable if and only if there exists a Lyapunov functions such that*

$$\begin{aligned} \alpha||x||_2^{r_1} \leq V(x) \leq \beta||x||_2^{r_2} \\ \dot{V}(x) \leq -\gamma||x||_2^{r_3}, \end{aligned} \tag{170}$$

where $\alpha, \beta, \gamma, r_1, r_2, r_3 \in \mathbb{R}_{>0}$ are constants with $r_3 > r_2$.

The decay rate in Equation (169) is shown in [6] where $k = r_1\left(\frac{r_3}{r_2} - 1\right)$, $\eta = \frac{r_2}{r_1}$, and $M > 0$ depends non-trivially on $\alpha, \beta, \gamma, r_1, r_2$, and r_3.

Note that the key difference between rational and exponential stability resides with the constants r_2 and r_3 from Equation (170) and c from Equation (167).

18.3 Structured Lyapunov Functions

In order to efficiently search for Lyapunov functions, it is natural to assign a template structure to the Lyapunov function and then perform an optimisation over the parameters. The structure chosen in this Thesis is for the Lyapunov functions to be polynomials and continuous piecewise-polynomials. This choice is not novel and the (non-)existence results presented below are, with one exception, all for polynomial Lyapunov functions.

First, a result by [1] states that there exist globally asymptotically stable polynomial vector fields without any polynomial Lyapunov function to prove it. In [1] they prove this with an explicit, surprisingly simple, two-dimensional system given as

$$\begin{aligned} \dot{x} &= -x + xy \\ \dot{y} &= -y. \end{aligned} \tag{171}$$

They then show that the system cannot admit to a polynomial Lyapunov function. They prove the stability property with a non-linear, non-polynomial C^∞ Lyapunov function, which then still is in accordance with Theorem 17. Interestingly, this example has a non-trivial Jacobian with negative eigenvalues. Thus the linearised system is exponentially stable and by Theorem 4.15 in [23] this is a necessary and sufficient condition for the equilibrium of the non-linear system to be locally exponentially stable. By the solution to the Lyapunov equation ([23]) there exists a local polynomial Lyapunov function showing the local exponential stability of the non-linear system (171).

Similar to the result above, [6] constructs a globally asymptotically stable polynomial vector field without any polynomial Lyapunov function to prove it. The difference however, is that this system has a trivial Jacobian. Thus linearising the system does not offer any insight into the stability properties. The proof of the non-existence hinges on the fact that the constructed vector field has algebraic irrational coefficients, and restricting the analysis to local asymptotic stability does not alleviate the non-existence of a polynomial Lyapunov function.

Considering local exponential stability, [38] proves that the existence of a polynomial Lyapunov function fulfilling Equation (167) is a necessary and sufficient condition for (sufficiently smooth) non-linear vector fields to be locally exponentially stable. The smoothness criterion is trivially fulfilled for polynomial vector fields. As such, when analysing for local exponential stability, limiting the search to polynomial Lyapunov functions is not conservative. This proof uses the Weierstrass approximation theorem which naturally disables a generalisation to global analysis.

Still considering local exponential stability, [16] proves that the existence of a continuous piecewise-affine Lyapunov function fulfilling Equation (167) is a necessary and sufficient condition for (sufficiently smooth) non-linear vector fields to be locally exponentially stable. The smoothness criterion is trivially fulfilled for polynomial vector fields. This result is interesting, since continuous piecewise-affine functions are a specific choice of continuous piecewise-polynomials.

During my PhD studies, I proved that the existence of a polynomial Lyapunov function fulfilling Equation (170) is a necessary and sufficient condition for (sufficiently smooth) non-linear vector fields to be locally rationally stable. The smoothness criterion is trivially fulfilled for polynomial vector fields. As such, when analysing for local rational stability, limiting the search to polynomial Lyapunov functions is not conservative. This result was reported in [28] and Appendix E contains the proof. It is inspired by the proof in [38] and once again the use of the Weierstrass approximation theorem disables a global analysis.

The interest in these negative and affirmative results is sparked by the effort to move the current state of Lyapunov theory from "existence is necessary and sufficient" to "if stable, this algorithm will show it". Numerical methods found their way into stability analysis as a way to overcome the necessity of experience and trial-and-error, and to aid the analysis of systems too comprehensive for hand calculations. However, so far all implementations suffer from only being sufficient, and if an algorithm fails it does not indicate instability. In Chapter Instability Certification, this issue is addressed further.

In light of the results above, local rational and exponential stability will

always admit to polynomial Lyapunov functions. For systems with irrational coefficients, local asymptotic stability can require a non-polynomial Lyapunov function as seen above. The difference between local asymptotic and local rational stability seems to be anchored somewhere between either allowing for vector fields with irrational coefficients, allowing for analytic vector fields, or only allowing for rational coefficient, see proposition 5.2 and 5.4 in [6]. It is beyond the scope of this Thesis to try to characterise the difference, and since the object is to devise numerical methods, these subtle differences could not be taken into account anyway. When analysing an unknown polynomial vector field, this provides a solid foundation for the choice of using polynomial Lyapunov functions.

Finally, I will add that by nature of the local description in the Bernstein basis, stability analysis using Lemma 18 and Lemma 19 can never address the question of global stability. While some authors seem to favour global analysis, I will argue for the contrary. When analysing a dynamical system with unknown stability properties, the number of equilibrium points will also be unknown. Should there be more than one equilibrium point, none of them can be globally asymptotically stable and thus applying global analysis will inherently fail.

Stability Certification

This Chapter contains the culmination of my studies. It transforms Lemma 18 and Lemma 19 to a linear feasibility problem, the solution of which is a Lyapunov function. When faced with infeasibility, I searched for an intelligent way to modify the problem. Although stability analysis differs from positivity certification, the insight obtained in Section 12.2 aided my search quite a lot. The solution turned out to be a specific utilisation of the information obtained from Farkas' Lemma.

For completeness, this Chapter concludes with a section covering an alternative method of utilising the Bernstein basis for synthesising Lyapunov functions. This method relies on some more properties of the basis polynomials, instead of the sign on the coefficients.

19 Linear Program for Synthesising

Lemma 18 enables the possibility of checking whether or not a known function $V(x)$ can certify the stability of an equilibrium of a known polynomial vector field $f(x)$ as given in Equation (152).

In this Section, the reverse question is considered: Design or synthesise a Lyapunov function showing the stability of a known polynomial vector field. To be consistent with the notion of a *certificate of positivity*, I shall reference the existence of a Lyapunov function as a *certificate of stability*.

Written explicitly in the Bernstein basis on a collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ a vector field reads

$$f(x) = \begin{bmatrix} f_1(x) = \sum_{|\alpha|=D} b(f_1, D, \sigma^i)\mathcal{B}_\alpha^D(\sigma^i) & \forall x \in \sigma^i, i \in \{1, 2, \ldots, m\} \\ f_2(x) = \sum_{|\alpha|=D} b(f_2, D, \sigma^i)\mathcal{B}_\alpha^D(\sigma^i) & \forall x \in \sigma^i, i \in \{1, 2, \ldots, m\} \\ \cdots & \\ f_n(x) = \sum_{|\alpha|=D} b(f_n, D, \sigma^i)\mathcal{B}_\alpha^D(\sigma^i) & \forall x \in \sigma^i, i \in \{1, 2, \ldots, m\} \end{bmatrix}, \tag{172}$$

where the degree in the description of f_i is D such that

$$D \geq \max_{i \in \{1,\dots,n\}} d_i, \tag{173}$$

where d_i is the actual degree of f_i. In the rest of this Thesis D is chosen to be $D = \max_{i \in \{1,\dots,n\}} d_i$.

Considering $V(x)$ and $\dot{V}(x)$ in Lemma 18 as unknown, there are two vectors of unknown coefficients, CV and CL. They are linked in such a way that CL are the coefficients after differentiation along trajectories of $f(x)$. To derive their relation, define the polynomial Lyapunov function $V(x)$ of degree d_V on the collection of simplices $K = \{\sigma^1, \sigma^2, \dots, \sigma^m\}$ according to Definition 6 as

$$V(x) = CV^i \mathcal{B}^{d_V}(\sigma^i) \qquad \forall x \in \sigma^i, i \in \{1, 2, \dots, m\}, \tag{174}$$

$$CV^{i^T} = {}^iB_j \, CV^{j^T} \qquad \forall\, i, j \in \{1, \dots, m\}. \tag{175}$$

To obtain the Lie derivative, first the partial derivative of $V(x)$ is needed. By Equation (135)

$$\frac{\partial V}{\partial x} = \sum_{|\tilde{\alpha}|=d_V-1} d_V [CV^i_{\tilde{\alpha}+e_0}, \dots, CV^i_{\tilde{\alpha}+e_n}] \varsigma^i \mathcal{B}^{d_V-1}_{\tilde{\alpha}}(\sigma^i) \quad \forall x \in \sigma^i, i \in \{1, 2, \dots, m\} \tag{176}$$

where ς^i is the derivative of the barycentric coordinates of the i^{th} simplex, see Equation (127). The coefficients of the j^{th} partial derivative are then identified as

$$b\left(\frac{\partial V}{\partial x_j}, d_V - 1, \sigma^i\right) = \begin{bmatrix} d_V [CV^i_{\tilde{\alpha}_1+e_0}, \cdots, CV^i_{\tilde{\alpha}_1+e_n}] \varsigma^i(:, j) \\ d_V [CV^i_{\tilde{\alpha}_2+e_0}, \cdots, CV^i_{\tilde{\alpha}_2+e_n}] \varsigma^i(:, j) \\ \cdots \\ d_V [CV^i_{\tilde{\alpha}_{N_{d_V-1}}+e_0}, \cdots, CV^i_{\tilde{\alpha}_{N_{d_V-1}}+e_n}] \varsigma^i(:, j) \end{bmatrix}^T, \tag{177}$$

where N_{d_V-1} is the number of α-combinations according to Equation (17) and $\varsigma^i(:, j)$ is the j^{th} column of ς^i.

The Lie derivative is obtained according to Equation (154) by multiplying the vector field Equation (172) with the partial derivative Equation (176). Multiplication is done according to Equation (114). For $\frac{\partial V}{\partial x_j}$ and $f_j(x)$ this gives

$$\begin{aligned} &b_{\hat{\alpha}}\left(\frac{\partial V}{\partial x_j} f_j(x), d_L, \sigma^i\right) = \\ &\sum_{\substack{|\hat{\alpha}-\alpha|=d_L \\ \hat{\alpha}-\alpha \geq 0}} b_\alpha(f_j, D, \sigma^i) b_{\hat{\alpha}-\alpha}\left(\frac{\partial V}{\partial x_j}, d_V - 1, \sigma^i\right) \frac{\binom{D}{\alpha}\binom{d_V-1}{\hat{\alpha}-\alpha}}{\binom{d_L}{\hat{\alpha}}}, \qquad \forall |\hat{\alpha}| = d_L \end{aligned} \tag{178}$$

and since the degree of $\frac{\partial V}{\partial x_j} f_j(x)$ is $d_L = d_V - 1 + D$ for all j, the coefficients of the Lie derivative are obtained directly by adding the coefficients according to Equation (118) as

$$b_{\hat{\alpha}}\left(\dot{V}, d_L, \sigma^i\right) = \sum_{j=1}^{n} b_{\hat{\alpha}}\left(\frac{\partial V}{\partial x_j} f_j(x), d_L, \sigma^i\right), \qquad \forall |\hat{\alpha}| = d_L, \forall i \in \{1, 2, \ldots, m\}. \tag{179}$$

Since both the vector field and the Lyapunov function are polynomials, then so is the Lie derivative. Following the discussion between Definition 5 and Definition 6, the coefficients can be collected in a single vector CL as

$$CL^i = b\left(\dot{V}, d_L, \sigma^i\right). \tag{180}$$

This eliminates duplicate coefficients on the faces in the description of the Lie derivative, and since all manipulations performed in this derivation are linear, the final relation is

$$CL^T = A\ CV^T. \tag{181}$$

Here, A is a matrix with entries determined by the degree of the Lyapunov function, by the conventions for numbering simplices and vertices from Section 3, the coefficients of the vector field being analysed, and the specific collection of simplices K on which all the polynomials involved are described on.

With this derivation, the synthesis problem becomes a linear feasibility problem as

$$\begin{aligned} \min_{CV} \quad & \\ \text{s.t.} \quad & l^x \leq CV^T \leq u^x \\ & l^c \leq A\ CV^T \leq u^c \end{aligned} \tag{182}$$

where l^x, u^x, l^c, and u^c are lower bound on design variable, upper bound on design variable, lover bound on constraints, and upper bound on constraints respectively.

The lower and upper bounds on the design variables are determined by Equations (161a) to (161d) such that $l_i^x \in \{0, 1\}$ and $u_i^x \in \{0, \infty\}$ depending on whether the i^{th} entry of CV corresponds to a coefficient at the vertex at the origin, to a coefficient at a vertex not at the origin, or to one of the remaining coefficients in the collection of simplices. This translates > 0 to ≥ 1. Since any positive scaling of a Lyapunov function remains a Lyapunov function, this does not change the feasibility of Equation (182).

Similarly, the lower and upper bounds on the constraints are determined by Equations (161e) to (161h) such that $l_i^c \in \{-\infty, 0\}$ and $u_i^c \in \{-1, 0\}$ depending on whether the i^{th} entry of CL corresponds to a coefficient at the vertex at the origin, to a coefficient at a vertex not at the origin, or to one of the remaining coefficients in the collection of simplices. This translates < 0 to ≤ -1 which again does not affect the feasibility.

The existence of a solution to Equation 182 is unaffected by the choice of an objective function. Throughout the rest of this Chapter, it is chosen to minimise the sum of the design variables, e.g. the 1-norm. Although this is a well-known heuristic for obtaining a spares solution, it is chosen simply to improve visual representation in the coming examples.

Completely analogously, a relation between CV and $b(\dot{V}_i, d_L, \sigma^i)$ can be obtained if instead the Lyapunov function is defined as a continuous piecewise-polynomial, according to Definition 7. This renders the Lie derivative a discontinuous piecewise-polynomial, as given in Definition 8. As mentioned after the definition, there are no duplicate coefficients on the faces, thus no apparent reason to combine the coefficient vectors $b(\dot{V}_i, d_L, \sigma^i)$ into one vector. This makes the relation read

$$b^T(\dot{V}_i, d_L, \sigma^i) = A_i \, CV^T \qquad \forall i \in \{1, 2, \ldots, m\}. \tag{183}$$

Here A_i is a matrix with entries determined by the degree of the Lyapunov function, by the conventions for numbering simplices and vertices from Section 3, the coefficients of the vector field being analysed, and the i^{th} simplex.

By abuse of notation, to obtain a relation like in Equation (181), let the vector of all coefficients defining the Lie derivative be denoted

$$CL = [b(\dot{V}_1, d_L, \sigma^1), b(\dot{V}_2, d_L, \sigma^2), \ldots, b(\dot{V}_m, d_L, \sigma^m)]. \tag{184}$$

This enables a linear feasibility program identical to Equation (182) with

$$CL^T = A \, CV^T, \tag{185}$$

where

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \cdots \\ A_m \end{bmatrix}. \tag{186}$$

In this case, the upper and lower bounds on design variables and constraints are determined according to Equations (165a) to (165h) in Lemma 19.

Henceforth, I will refer to the feasibility problem of both polynomial and continuous piecewise-polynomial Lyapunov functions by Equation (182) regardless of CL and A being defined by Equations (180) and (181) or by Equations (184) and (185). To ease the notation, I introduce the following.

Notation 25 *Let $f : U \to \mathbb{R}^n$, where $U \subseteq \mathbb{R}^n$ and $U_0 \subseteq U$ is a closed hypercube, be a polynomial vector field defining the dynamical system $\Sigma : \dot{x} = f(x)$. The linear feasibility problem defined by* (182) *for Σ on a collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ which covers U_0, is written $LP(\Sigma, K)$. To be read as "The linear synthesis problem for Σ, when V and f are defined on K".*

Here, the equilibrium point is assumed to be the origin and the collection of simplices K covering U_0 is obtained by applying Kuhn's triangulation on the facets of U_0 and adding the origin as a vertex to the simplices, see Definition 3. Whether $LP(\Sigma, K)$ synthesises a polynomial Lyapunov function or a continuous piecewise-polynomial Lyapunov function will be evident from context.

Notation 26 *A vector of coefficients CV is called feasible if it fulfils* (182)*, and $LP(\Sigma, K)$ is called solvable if a feasible CV exists. Otherwise, it is called unsolvable.*

If, for a given vector field $f(x)$, $LP(\Sigma, K)$ is solvable, then CV is a certificate of stability, which proves that the system is at least locally asymptotically stable. Further analysis of the synthesised Lyapunov function can assess if the Lyapunov function proves rational or exponential stability, or just asymptotic stability. This analysis is not considered in this Thesis. On the other hand, if $LP(\Sigma, K)$ is unsolvable, no information of the stability properties of the origin can be inferred. This situation is the focus of the rest of this Chapter. Later, an algorithm to refine the triangulation is derived, but first some examples.

Example

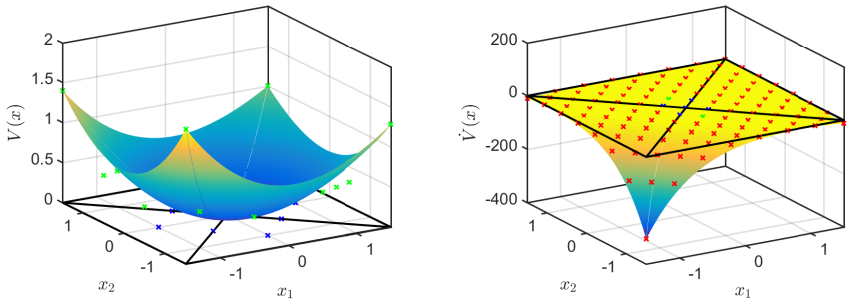
Consider the polynomial vector field $f : \mathbb{R}^2 \to \mathbb{R}^2$, taken from [2], given as

$$\begin{aligned} c &= \begin{bmatrix} -1 & 2 & -1 & 4 & -8 & 4 & -1 & 4 & 0 & -4 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & -9 & 10 & 0 & 2 & -8 & -4 & -1 & 4 & -4 \end{bmatrix} \\ \gamma &= \begin{bmatrix} 3 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 4 & 3 & 2 & 0 & 3 & 2 & 1 \end{bmatrix} \end{aligned} \tag{187}$$

where, by Equation (11), $f_1(x)$ and $f_2(x)$ are

$$\begin{aligned} f_1(x) = \sum_{k=1}^{13} c_{1,k} x^{\gamma_k} = &- x_1^3 x_2^2 + 2x_1^3 x_2 - x_1^3 + 4x_1^2 x_2^2 - 8x_1^2 x_2 \\ &+ 4x_1^2 - x_1 x_2^4 + 4x_1 x_2^3 - 4x_1 + 10x_2^2, \end{aligned} \tag{188}$$

$$\begin{aligned} f_2(x) = \sum_{k=1}^{13} c_{2,k} x^{\gamma_k} = &- 9x_1^2 x_2 + 10x_1^2 + 2x_1 x_2^3 - 8x_1 x_2^2 \\ &- 4x_1 - x_2^3 + 4x_2^2 - 4x_2. \end{aligned} \tag{189}$$



(a) Polynomial Lyapunov function for (187). (b) Polynomial Lie derivative for (187).

Figure 22: Solution to $LP(\Sigma, K)$ in the example. The solid lines are the facets of the simplices and the red, blue, and green crosses are the control points. The colour of the crosses indicate whether the coefficient is negative (red), positive (green), or equal to zero (blue). A few crosses on Figure (b) are green, i.e. positive. Their values are in the magnitude of 10^{-11} and depend on the accuracy of the solver.

Note that the description of a polynomial vector field in the monomial basis is equal to the description of a single polynomial, except the coefficient vector c from Equation (11) is expanded to a coefficient matrix as seen above.

In order to investigate the local stability of the origin using the linear synthesis problem $LP(\Sigma, K)$, two things must be specified. The degree and type of the Lyapunov function to search for and the hypercube to define it on. In this Example, a polynomial Lyapunov function of degree $d_V = 3$ is used and it is defined on the hypercube $U_0 = [\pm 1.5]^2$.

In this Thesis, linear programming problems are solved using the commercially available software MOSEK. It provides optimisation solutions for linear, quadratic, conic, and mixed integer problems, and the interested reader is encouraged to visit mosek.com for more information and to obtain a free academic trial. MOSEK was chosen due to some inherent features which will become evident and important later on.

Running the program reveals that $LP(\Sigma, K)$ is solvable and the synthesised Lyapunov function is

$$V(x) = -0.0322x_1^3 - 0.0165x_1^2x_2 + 0.298x_1^2 - 0.0271x_1x_2^2 - 0.0253x_2^3 + 0.298x_2^2 \tag{190}$$

with a Lie derivative

$$\begin{aligned}\dot{V}(x) =& 0.0966x_1^5x_2^2 - 0.193x_1^5x_2 + 0.0966x_1^5 + 0.033x_1^4x_2^3 - 1.05x_1^4x_2^2 + \\ & 2.15x_1^4x_2 - 1.15x_1^4 + 0.124x_1^3x_2^4 - 0.605x_1^3x_2^3 + 3.3x_1^3x_2^2 - \\ & 5.44x_1^3x_2 + 2.84x_1^3 + 0.033x_1^2x_2^5 - 0.945x_1^2x_2^4 + 3.74x_1^2x_2^3 - \\ & 7.27x_1^2x_2^2 + 6.38x_1^2x_2 - 2.38x_1^2 + 0.0271x_1x_2^6 - 0.26x_1x_2^5 + \\ & 1.85x_1x_2^4 - 5.32x_1x_2^3 + 6.59x_1x_2^2 - 2.38x_1x_2 + 0.076x_2^5 - \\ & 1.17x_2^4 + 2.69x_2^3 - 2.38x_2^2.\end{aligned} \tag{191}$$

Figure 22 shows the Lyapunov function and the Lie derivative on U_0.

■

Example

Consider the polynomial vector field $f : \mathbb{R}^2 \to \mathbb{R}^2$, taken from [42], given as

$$\begin{aligned} c &= \begin{bmatrix} -2 & 0 & -0.5 & -0.5 & 0 & 0 \\ 0 & 0.25 & -0.125 & 0 & 0.25 & -0.4125 \end{bmatrix} \\ \gamma &= \begin{bmatrix} 3 & 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 2 & 1 \end{bmatrix}. \end{aligned} \tag{192}$$

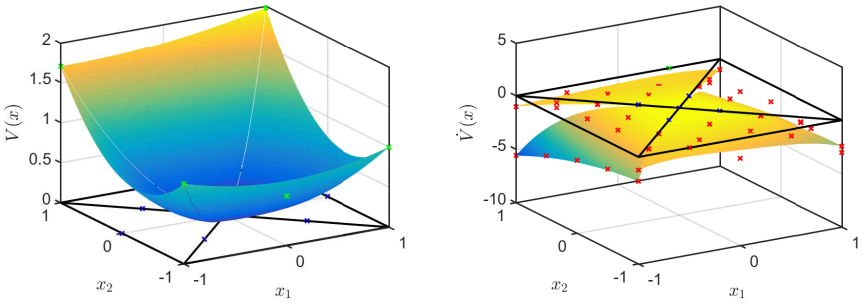
To investigate the local stability of the origin, consider the hypercube $U_0 = [\pm 1]^2$. Choosing to synthesise a polynomial Lyapunov function of degree $d_V = 2$ results in an $LP(\Sigma, K)$ which is unsolvable. In contrast, choosing to synthesise a continuous piecewise-polynomial Lyapunov function of degree $d_V = 2$ renders $LP(\Sigma, K)$ solvable. This is an example of the fact that continuous piecewise-polynomials belong to a larger class of functions which has polynomials as a subset.

The synthesised Lyapunov function is

$$V(x) = \begin{cases} V_1(x) = 0.746x_1^2 + 0.491x_1x_2 + 0.746x_2^2 & \forall x \in \sigma^1 \\ V_2(x) = 0.192x_1^2 + 0.808x_2^2 & \forall x \in \sigma^2 \\ V_3(x) = 0.676x_1^2 - 0.351x_1x_2 + 0.676x_2^2 & \forall x \in \sigma^3 \\ V_4(x) = 0.0738x_1^2 + 0.14x_1x_2 + 1.77x_2^2 & \forall x \in \sigma^4 \end{cases}, \tag{193}$$

with a Lie derivative

$$\dot{V}(x) = \begin{cases} \dot{V}_1(x) & \forall x \in \sigma^1 \\ \dot{V}_2(x) & \forall x \in \sigma^2 \\ \dot{V}_3(x) & \forall x \in \sigma^3 \\ \dot{V}_4(x) & \forall x \in \sigma^4 \end{cases}, \tag{194}$$



(a) Continuous piecewise-polynomial Lyapunov function for (192).

(b) Discontinuous piecewise-polynomial Lie derivative for (192).

Figure 23: Solution to $LP(\Sigma, K)$ in the example. Note the discontinuity of the Lie derivative on the facets. One cross on Figure (b) is green but its value is in the magnitude of 10^{-10}.

where

$$\begin{aligned}\dot{V}_1(x) = &-0.295x_1^4 - 0.28x_1^3x_2 + 0.0349x_1^2x_2^2 - 0.0913x_1^2x_2 - 0.0738x_1^2 \\ &+ 0.884x_1x_2^3 - 0.477x_1x_2^2 - 0.128x_1x_2 + 0.884x_2^3 - 1.46x_2^2 \end{aligned} \tag{195}$$

$$\begin{aligned}\dot{V}_2(x) = &-2.7x_1^4 + 0.702x_1^3x_2 - 0.0878x_1^2x_2^2 - 0.632x_1^2x_2 - 0.676x_1^2 \\ &+ 0.338x_1x_2^3 - 0.0811x_1x_2^2 + 0.32x_1x_2 + 0.338x_2^3 - 0.557x_2^2 \end{aligned} \tag{196}$$

$$\begin{aligned}\dot{V}_3(x) = &-0.768x_1^4 - 0.192x_1^2x_2 - 0.192x_1^2 + 0.404x_1x_2^3 - 0.202x_1x_2^2 \\ &+ 0.404x_2^3 - 0.667x_2^2 \end{aligned} \tag{197}$$

$$\begin{aligned}\dot{V}_4(x) = &-2.98x_1^4 - 0.982x_1^3x_2 + 0.123x_1^2x_2^2 - 0.807x_1^2x_2 - 0.746x_1^2 \\ &+ 0.373x_1x_2^3 - 0.309x_1x_2^2 - 0.448x_1x_2 + 0.373x_2^3 - 0.615x_2^2. \end{aligned} \tag{198}$$

Figure 23 shows the Lyapunov function and the Lie derivative on U_0.

■

20 Infeasibility

Having seen examples of when everything works and $LP(\Sigma, K)$ is solvable on the initial collection of simplices, the following two sections consider strategies for modifying the collection in an attempt to obtain a solvable $LP(\Sigma, K)$. This is motivated by the experience obtained in Chapter Positivity Certification, particularly the fact that if a Lyapunov function exists, $LP(\Sigma, K)$ may be unsolvable simply due to the particular collection of simplices K. Sub-dividing the simplices into a new collection of simplices K^* may render $LP(\Sigma, K^*)$ solvable.

To be consistent with the sub-division strategies for certifying positivity being split into pre-defined and adaptive vertex placement, the first section considers generic methods and the second section considers methods for utilising information from the current problem.

20.1 Regular Sub-Division

In this Section, the collection of simplices is sought modified by applying a regular sub-division routine to all simplices in the collection. Remember that a regular sub-division routine was any sub-division routine with a constant shrinking factor C such that $0 < C < 1$, as introduced in Section 12. Two such routines are the standard triangulation and the binary splitting.

To distinguish between different collections of simplices, denote the initial collection obtained by Kuhn's Triangulation as K_0 and define the following.

Definition 27 *Let K_0 be a collection of simplices covering the hypercube U_0 obtained from Kuhn's triangulation, see Definition 3. Let S be a regular sub-division routine. Then*

$$K_1 = S(K_0) \tag{199}$$

is the collection of simplices after 1 application of the sub-division routine on all simplices in the collection K_0. Then

$$K_{n+1} = S(K_n) \tag{200}$$

is recursively defined as the collection of simplices after n application of the sub-division routine on all simplices in all intermediate collections of simplices K_0, K_1 to K_n.

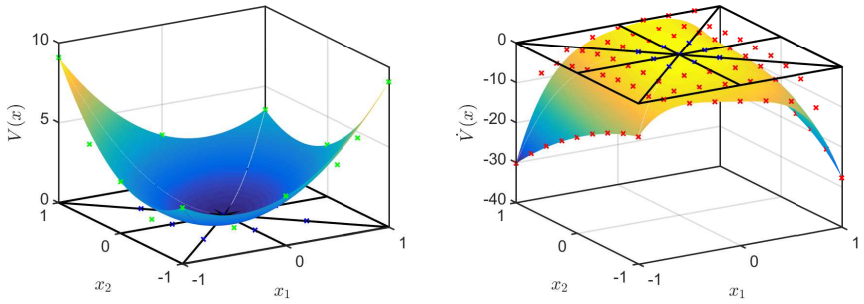
This enables the following algorithm.

Algorithm 28 *(The BBAlgorithm [29])*
Input: *Dynamical system Σ, closed hypercube U_0, and type and degree of the Lyapunov function ($d_V \geq 1$).*
Output: *Triangulation of U_0 and Lyapunov function V defined on the resulting simplices.*
Procedure:

0) Set iteration counter $k = 1$ and get initial $K^{\{k\}} = K_0$.

1) If $LP(\Sigma, K^{\{k\}})$ is solvable, then return $K^{\{k\}}$ and CV. Otherwise, go to 2).

2) Use a regular sub-division routine on all simplices to get $K^{\{k+1\}} = S\left(K^{\{k\}}\right)$ and set $k = k + 1$. Go to 1).



(a) Polynomial Lyapunov function for (192). (b) Polynomial Lie derivative for (192).

Figure 24: Solution to $LP(\Sigma, K)$ obtained after one sub-division using binary splitting on the initial collection of simplices. Compare to Figure 23 for the solution when synthesising a continuous piecewise-polynomial Lyapunov function for the vector field in Equation (192).

Initial work comparing standard triangulation and binary splitting to obtain a collection of simplices K^* rendering $LP(\Sigma, K^*)$ solvable did not reveal any significant advantage of either. Considerations regarding bookkeeping when implementing the routines in arbitrary dimension led to a choice of utilising the binary splitting in the following. Readers interested in standard triangulation in arbitrary dimension are referred to [17].

Example

In this Example, the system from the previous example, Equation (192), is revisited. Since $LP(\Sigma, K)$ was unsolvable when synthesising a polynomial Lyapunov function of degree $d_V = 2$, the BBAlgorithm utilising binary splitting in step 2 is used for the stability analysis. In iteration two, after applying binary splitting once, $LP(\Sigma, K^{\{2\}})$ is solvable. The synthesised Lyapunov function and the Lie derivative are shown in Figure 24 on $U_0 = [\pm 1]^2$ from above, but now on a collection of eight simplices. The Lyapunov function is

$$V(x) = 3.22x_1^2 - 2.78x_1x_2 + 3.08x_2^2 \tag{201}$$

with a Lie derivative

$$\begin{aligned}\dot{V}(x) = &-12.9x_1^4 + 5.56x_1^3x_2 - 0.695x_1^2x_2^2 - 2.87x_1^2x_2 - 3.22x_1^2 \\ &+ 1.54x_1x_2^3 - 0.0744x_1x_2^2 + 2.54x_1x_2 + 1.54x_2^3 - 2.54x_2^2.\end{aligned} \tag{202}$$

■

Existence and Solvability

In this Section, I pinpoint an inherent drawback with the BBAlgorithm and using the Bernstein basis in general, and then argue that the drawback may not be as profound as it could seem. First, recall the notion of a Bernstein basis certifying collection as a collection of simplices K such that, for a given positive definite polynomial p, $C \geq 0$ when p is described on K. Then p was said to be Bernstein basis certifiable. See Notation 12 for details.

Theorem 10 is the foundation for certifying positivity of positive polynomials using sub-division. Using the theorem as justification to synthesise positive definite polynomials does not encompass bi-implication. As seen from the counterexample in Appendix D, positive definite polynomials which are not Bernstein basis certifiable do exist. Such a polynomial can never be the solution to any $LP(\Sigma, K)$.

Seeing as Lyapunov functions possess a certain structure, is was of interest to investigate if the counterexample could serve as a Lyapunov function for a stable vector field. To this end, a stable vector field was synthesised such that it admitted the counterexample as a Lyapunov function. As such, this system proves the existence of stable vector fields which admit to polynomial Lyapunov functions without any Bernstein basis certifying collections. A natural question was whether the synthesised systems admitted to polynomial Bernstein basis certifiable Lyapunov functions. The system was analysed using the BBAlgorithm. It retuned a feasible CV shoving that the system indeed did admit to polynomial Bernstein basis certifiable Lyapunov functions. See Appendix D for the full analysis.

This result exemplifies a desirable feature of Lyapunov analysis. When interested in assessing the stability properties of a given system, the existence of Lyapunov functions and the ability of the BBAlgorithm to find at least one of them is enough. Whether or not there exist Lyapunov functions which the BBAlgorithm cannot find does not matter, as long as there exists at least one Lyapunov functions which the BBAlgorithm can find.

Naturally, the work on converse Lyapunov theorems has never considered whether or not the necessary and sufficient existence of a polynomial Lyapunov function also ensures the existence of a polynomial Lyapunov function which is Bernsten basis certifiable. To the best of my knowledge, there are no results in this regard on the characterisation of positive definite polynomials. Such a characterisation could enable a result on whether the existence of a polynomial Lyapunov function would also ensure the existence of a polynomial Bernstein basis certifiable Lyapunov function. This was certainly the case for the synthesised vector field in Appendix D. Despite my best efforts, I have not been able to determine the underlying mechanism behind the non-existence of a Bernstein basis certifying collection for the counterexample, let alone begun to attack the problem in general. Such a characterisation could

possibly remove the (apparent) conservatism introduced in the analysis when using the BBAlgorithm.

Convergence

In [29] a convergence result is proven under assumptions regarding the existence of a Bernstein basis certifying collection. Further work attempting to loosen the assumptions, has shown that the assumptions were in fact not restrictive enough.

The collection obtained in step 2 of the BBAlgorithm cannot consist of arbitrary simplices. The simplices created in step 2 depend on the hypercube and on which sub-division routine is used. For instance, for a two dimensional hypercube symmetric around the origin using binary splitting, all simplices at any iteration are identical down to scaling and rotation, see Figure 25(a) below. This calls for the following definitions.

Definition 29 *(S-Generated Simplex) For the collections of simplices K_0 to K_N obtained by applying the sub-division routine S up to N times, if*

$$\sigma \in K_n \tag{203}$$

for some $n \in \{1, 2, \ldots, N\}$, then σ is an S-generated simplex of level n. Then

$$\mathcal{K}_N = \bigcup_{n=0}^{N} K_n \tag{204}$$

is the set of all S-generated simplices of at most level N, and if

$$\sigma \in \mathcal{K}_N \tag{205}$$

σ is an S-generated simplex of at most level N.

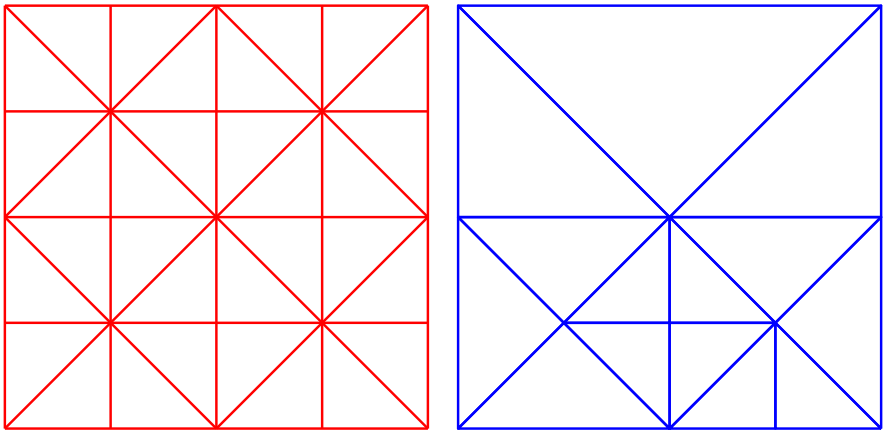
Definition 30 *(S-Generated Collection) Let $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ be a collection of simplices. For a given hypercube and sub-division routine S, if*

$$\sigma^i \in \mathcal{K}_N \qquad \forall i \in \{1, 2, \ldots, m\} \tag{206}$$

then K is an S-Generated Collection. The simplices in K are all S-generated simplices of at most level N.

For an example of *S*-generated simplices, consider Figure 25(a) where a two dimensional hypercube symmetric around the origin is sub-divided three times using binary splitting, resulting in the *S*-generated collection of simplices $K_3 = \{\sigma^1, \sigma^2, \ldots, \sigma^{32}\}$. All simplices in K_3 are level 3 simplices, leading to

$$\sigma^i \in K_3 \qquad \forall i \in \{1, 2, \ldots, 32\}. \tag{207}$$



(a) S-generated simplices. Here the sub-division routine is the binary splitting and $K_3 = \{\sigma^1, \sigma^2, \ldots, \sigma^{32}\}$

(b) S-generated collection. Here $K^* = \{\sigma^{*1}, \sigma^{*2}, \ldots, \sigma^{*14}\}$

Figure 25: Clarification of the notation of S-generated simplices and S-generated collections.

More generally, consider Figure 25(b) where the same two dimensional hypercube symmetric around the origin is covered by the S-generated collection $K^* = \{\sigma^{*1}, \sigma^{*2}, \ldots, \sigma^{*14}\}$. In this case

$$\sigma^{*i} \in \mathcal{K}_3 \qquad \forall i \in \{1, 2, \ldots, 14\} \tag{208}$$

meaning that the simplices in K^* are of level 3 at most.

It is possible to prove a convergence result for the BBAlgorithm with this notation. First, the notion of a Bernstein basis certifying collection is expanded to Lyapunov functions as follows.

Notation 31 *(Lyapunov Bernstein Basis Certifying Collection) Let V be a Lyapunov function and $\dot{V}$ its Lie derivative for a given dynamical system Σ. The collection K is said to be a Lyapunov Bernstein basis certifying collection for V, if it is a Bernstein basis certifying collection for V and $-\dot{V}$.*

Before stating the convergence result, a lemma is needed.

Lemma 32 *([29]) Let the polynomial p of degree d be defined in the Bernstein basis of degree $D \geq d$ on a simplex $\hat{\sigma}$, and let $b(p, D, \hat{\sigma}) \geq 0$. Then any sub-division of $\hat{\sigma}$ into a collection of simplices $K = \{\sigma^1, \sigma^2, \ldots, \sigma^m\}$ such that*

$$\hat{\sigma} = \bigcup_{\sigma \in K} \sigma, \tag{209}$$

will preserve $b(p, D, \sigma^i) \geq 0$, $\forall i \in \{1, \ldots, m\}$.

Proof
This follows from the fact that $b(p, D, \sigma^i)$ are calculated as convex combinations of $b(p, D, \hat{\sigma})$. See [26] for details.

■

The convergence result reads as follows.

Proposition 33 *Let V^* be a polynomial Lyapunov function of degree less than or equal to the degree input to Algorithm 28 for the locally asymptotically stable system Σ. Assume the existence of an S-generated Lyapunov Bernstein basis certifying collection K^* for V^*. Then Algorithm 28 converges to a collection K' making $LP(\Sigma, K')$ solvable in finitely many steps, and the solution V' is a Lyapunov function for Σ.*

Proof
Let $K^* = \{\sigma^{*1}, \sigma^{*2}, \ldots, \sigma^{*m^*}\}$ be the S-generated Lyapunov Bernstein basis certifying collection for V^*. Let N be such that

$$\sigma^{*i} \in \mathcal{K}_N \qquad \forall i \in \{1, 2, \ldots, m^*\}, \tag{210}$$

e.g. all simplices in K^* are of at most level N. Then N sub-divisions of K_0 result in the collection K_N where all simplices are of level N. Setting $K' = K_N$, it follows from Lemma 32 that the coefficients of V' when described on K' are $CV' \geq 0$, since the coefficients of V^* when described on K^*, by assumption, are $CV^* \geq 0$. This makes $LP(\Sigma, K^{\{N\}})$ solvable.

Prior to obtaining the collection K_N, it may happen at iteration $k < N$ that $LP(\Sigma, K^{\{k\}})$ is solvable. In this case $K' = K^{\{k\}}$ and the algorithm has still converged.

■

Regarding the choice of which degree of the Lyapunov function d_V to search for, one argument is to choose a very high degree in an attempt to fulfil $d_{V^*} \leq d_V$. However, this will lead to $LP(\Sigma, K)$'s with a (possibly unnecessary) large number of variables and constraints, to an increase in runtime, and possibly challenge the capabilities of the hardware on which $LP(\Sigma, K)$ is sought solved. On the other hand, choosing a degree too small might result in an unsolvable $LP(\Sigma, K)$ simply because $d_{V^*} \leq d_V$ is violated.

Engineering intuition and experience from countless examples leaves me with the rule of thumb that a good choice of d_V is $d_V \approx D$, where D is the maximal degree of the polynomials in the vector field being analysed, see Equation (173).

Despite Proposition 33 being mathematically sound, is it not very encouraging in terms of engineering. The assumption of an S-generated Lyapunov Bernstein basis certifying collection seems very restrictive. On top of that, there is no way around it; sub-division routines can only generate S-generated simplices.

One way to alleviate the restrictive nature of the assumption, would be to obtain a result regarding robustness in the existence of Bernstein basis certifying collections. This again involves a characterisation of positive definite polynomials without Bernstein basis certifying collections, and their ability to serve as Lyapunov functions. This is an open question.

If the BBAlgorithm is used to analyse an unstable vector field, no collection of simplices will ever make $LP(\Sigma, K)$ solvable and the program will never terminate. This is an undesirable feature, which is addressed in the next chapter. Before that, the next section considers the utilisation of information of the current problem, to guide the sub-division.

20.2 Irregular Sub-Division

Certifying positivity and certifying stability are similar in the sense that a coefficient vector of a known polynomial with a negative entry does not imply that the polynomial cannot be positive, and that an unsolvable $LP(\Sigma, K)$ does not imply that the system cannot be stable. They differ, however, in that a coefficient vector with negative entries could be utilised to guide the placement of the next vertex, and that an unsolvable $LP(\Sigma, K)$ does not offer any information regarding where to place new vertices.

Inspired by the insight obtained into *where* to place new vertices when certifying positivity in Section 12.2, I investigated different strategies on how to incorporate information from a given vector field into the sub-division routine. I was interested in obtaining a notion similar to the most negative coefficient, as used in the grid point triangulation.

Initially, both the design variables and the constraints were of interest. By Equation (27), the location of a design variable, which is a coefficient of the Lyapunov function if $LP(\Sigma, K)$ is solvable, is located somewhere in the hypercube. The same equation determines the location of the constraints. I quickly realised that the design variables were of no use in this endeavour, since only the constraints contain information regarding the vector field being analysed. Thus it was the constraints and their location in the hypercube which somehow could be used to guide the placement of new vertices. But could one talk about the most negative constraint? The constraint most responsible for the infeasibility of $LP(\Sigma, K)$?

I pondered this question for quite some time, eventually publishing [27] on the use of slack variables to identify the violated constraints in an artificial solution to an augmented linear problem. My first attempt was, for each simplex with one or more violated constraints, to place a new vertex at the location of the most violated constraint. This would then be completely reminiscent of the most negative coefficient from positivity certification.

In some regard, it turned out that this strategy was a bit too ambitious.

Examples showed that trying to target specific constraints the way grid point triangulation targeted specific coefficients, did not work as intended. In a given simplex with violated constraints, the location of the most violated constraint simply did not correspond to the most opportune location to place the next vertex. However, the examples did reveal that the strategy successfully targeted simplices with violated constraints as a whole, and did not sub-divide simplices without any violated constraints. This allowed for a tailored sub-division routine, based on information from the vector field being analysed where only simplices with constraints contributing to the unsolvability of $LP(\Sigma, K)$ were sub-divided by a regular sub-division routine.

After the work on [27], I felt like the use of slack variables was a crude solution. I felt like there could be a more elegant way of identifying the constraints responsible for the unsolvability. The answer revealed itself in the form of Farkas' lemma.

Lemma 34 *(Farkas' Lemma [11]) Given a matrix $\bar{A}$ and a vector b exactly one of the following two propositions is true:*

1. $\exists x \ : \ \bar{A}x \leq b$,
2. $\exists y \ : \ y \geq 0, \ \bar{A}^T y = 0, \ b^T y < 0$.

That is, if the proposition 1. is infeasible, then there exists a y certifying the infeasibility. This is interesting, since the non-zero elements of y identify some or all of the constraints responsible for the infeasibility [4].

The reason why the certificate of infeasibility does not necessarily identify all constraints responsible for the infeasibility, is because the certificate y can fulfil proposition 2. without having non-zero entries corresponding to all constraints responsible for the infeasibility. To see this, consider the following system of inequalities.

$$x_1 \leq 0 \tag{211a}$$
$$-x_1 \leq -1 \tag{211b}$$
$$x_2 \leq 0 \tag{211c}$$
$$-x_2 \leq -1. \tag{211d}$$

Here Equation (211a) and Equation (211b) constitutes an unsolvable subset, and so does Equation (211c) and Equation (211d). Identifying either, is enough to certify the unsolvability of Equation (211). Thus e.g. $y_1 = [1, 1, 0, 0]^T$ and $y_2 = [0, 0, 1, 1]^T$ are both certificates of infeasibility of Equation (211).

Note that when solving linear programs using either simplex or dual-simplex methods, the certificate of infeasibility is a so-called basis certificate [3]. When solving using an interior-point algorithm, the obtained certificate

is a linear combination of all basis certificates, but the basis certificates can be recovered using the methods described in [3]. These methods are not detailed here, and henceforth when referring to a certificate of infeasibility it is assumed to be a basis certificate.

Simple manipulations can transform the linear feasibility problem from Equation (182) into the system of inequalities in proposition 1 in Farkas' Lemma. This transformation is included for completeness. The constraints and bounds on the design variables in Equation (182) were

$$\begin{aligned} l^x \leq & CV^T \leq u^x \\ l^c \leq & A\ CV^T \leq u^c \end{aligned} \tag{212}$$

which can be rewritten as

$$\begin{aligned} b_1 &= \begin{bmatrix} l^x \\ -u^x \end{bmatrix} \leq \begin{bmatrix} CV^T \\ -CV^T \end{bmatrix} = z \\ \tilde{A}z &= \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} CV^T \\ -CV^T \end{bmatrix} \leq \begin{bmatrix} u^c \\ -l^c \end{bmatrix} = b_2. \end{aligned} \tag{213}$$

Combining yields

$$\bar{A}z = \begin{bmatrix} -I \\ \tilde{A} \end{bmatrix} z \leq \begin{bmatrix} -b_1 \\ b_2 \end{bmatrix} = b. \tag{214}$$

At the beginning of my studies, there were no considerations regarding a choice of software for solving linear programming problems. After Farkas' lemma caught my attention, the MOSEK software naturally stood out as advantageous. When asked to solve an infeasible problem, MOSEK automatically returns a certificate of infeasibility.

To identify not only constraints in this way, but the simplices which they belong to, the maps Δ^i from Equations (52) and (53) can be used to identify the simplices identified by the certificate of infeasibility y. To this end, define the following operator.

Definition 35 *Let $LP(\Sigma, K)$ be infeasible where the collection of simplices has m simplices. Let $y \in \mathbb{R}^{N_y}_{\geq 0}$ be a certificate of infeasibility with N_y entries corresponding to $LP(\Sigma, K)$ having N_y inequalities. Define $\Gamma : \mathbb{R}^{N_y} \to \{0,1\}^m$ as a map from constraints to simplices such that*

$$\Gamma(y) = Y = (Y_1, Y_2, \ldots, Y_m) \tag{215}$$

where

$$Y_i = \begin{cases} 0, & \text{if simplex } i \text{ had no constraints identified by } y \\ 1, & \text{if simplex } i \text{ had one or more constraints identified by } y. \end{cases} \tag{216}$$

This allows for the following method for obtaining a sub-division where simplices are sub-divided by a regular sub-division routine, but where only some simplices are sub-divided at each iteration.

Notation 36 *(Certificate of Infeasibility Identifying (CII) Method) When $LP(\Sigma, K)$ is unsolvable, use Farkas' lemma to get a certificate of infeasibility y to identify, possibly a subset, of the constraints responsible for the infeasibility. When $y_i > 0$ denote the i^{th} constraint as a CII constraint. When $Y_i = 1$ denote the i^{th} simplex as a CII simplex and let K_Y denote all CII simplices in the collection K.*

The BBAlgorithm is modified using the certificate of infeasibility identifying method in step 2, such that it becomes the following.

Algorithm 37 *(Modified BBAlgorithm [29])*
Input: *Dynamical system Σ, closed hypercube U_0, and type and degree of the Lyapunov function ($d_V \geq 1$).*
Output: *Triangulation of U_0 and Lyapunov function V defined on the resulting simplices.*
Procedure:

0) Set iteration counter $k = 1$ and get initial $K^{\{k\}} = K_0$.

1) If $LP(\Sigma, K^{\{k\}})$ is solvable, then return $K^{\{k\}}$ and CV. Otherwise, go to 2).

2) Use a regular sub-division routine on CII simplices to get $K^{\{k+1\}} = S\left(K_Y^{\{k\}}\right)$ and set $k = k + 1$. Go to 1).

As for the BBAlgorithm, the modified BBAlgorithm will employ the binary splitting as the regular sub-division routine.

Example

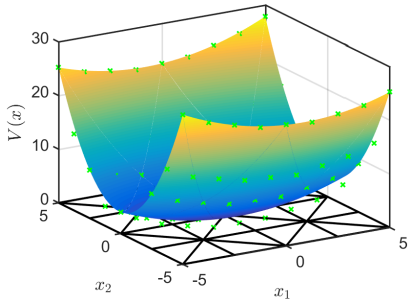
This example considers a polynomial vector field from [42] given by

$$\begin{aligned} c &= \begin{bmatrix} -1.5 & 0 & -1 & 0.5 & 0.5 & -2 & 1 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \gamma &= \begin{bmatrix} 1 & 0 & 2 & 1 & 0 & 3 & 2 \\ 0 & 1 & 0 & 1 & 2 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{217}$$

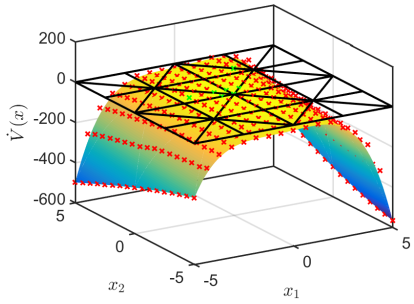
To clearly exemplify the difference between sub-dividing all simplices or only the CII simplices, the vector field is analysed using a polynomial Lyapunov function of degree $d_V = 2$ on the hypercube $U_0 = [\pm 5]^2$ using both the BBAlgorithm and the modified BBAlgorithm.

The synthesised Lyapunov function using the BBAlgorithm, i.e. binary splitting on all simplices, and its Lie derivative are shown in Figure 26(a) and Figure 26(b). The Lyapunov function is

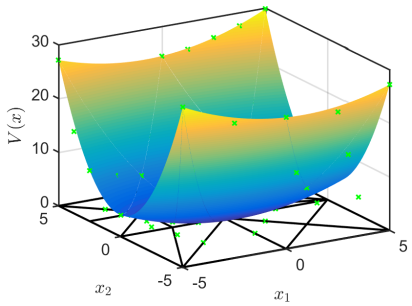
$$V(x) = 0.16x_1^2 + 0.0533x_1x_2 + 0.902x_2^2 \tag{218}$$



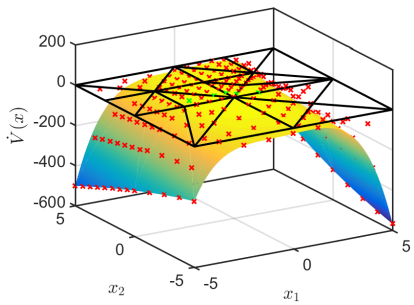
(a) Synthesised polynomial Lyapunov function V using the BBAlgorithm.



(b) The polynomial Lie derivative $\dot{V}$ using the BBAlgorithm.



(c) Synthesised polynomial Lyapunov function V using the modified BBAlgorithm.



(d) The polynomial Lie derivative $\dot{V}$ using the modified BBAlgorithm.

Figure 26: Difference between using binary splitting on all simplices or on the CII simplices. In both cases the synthesised Lyapunov function is a polynomial.

with a Lie derivative

$$\begin{aligned}\dot{V}(x) = &-0.64x_1^4 + 0.213x_1^3x_2 - 0.32x_1^3 + 0.0533x_1^2x_2^2 + 0.107x_1^2x_2 \\ &- 0.48x_1^2 + 0.187x_1x_2^2 - 0.107x_1x_2 + 0.0267x_2^3 - 0.902x_2^2.\end{aligned} \tag{219}$$

The synthesised polynomial Lyapunov function using the modified BBAlgorithm, i.e. binary splitting on the CII simplices, and its Lie derivative are shown in Figure 26(c) and Figure 26(d). The Lyapunov function is

$$V(x) = 0.16x_1^2 + 0.0533x_1x_2 + 0.98x_2^2 \tag{220}$$

with a Lie derivative

$$\begin{aligned}\dot{V}(x) = &-0.64x_1^4 + 0.213x_1^3x_2 - 0.32x_1^3 + 0.0533x_1^2x_2^2 + 0.107x_1^2x_2 \\ &- 0.48x_1^2 + 0.187x_1x_2^2 - 0.107x_1x_2 + 0.0267x_2^3 - 0.98x_2^2.\end{aligned} \tag{221}$$

Note how remarkably similar the two synthesised Lyapunov functions are.

Comparing the two resulting collections of simplices in Figure 26, it is clear that the CII method is able to tailor the collection and thus $LP(\Sigma, K)$ to the vector field being analysed. Comparing the two solvable $LP(\Sigma, K)$'s where the collection K has been obtained by either method, the resulting linear program is smaller for the CII method than the resulting linear program when sub-dividing all simplices. The solvable $LP(\Sigma, K)$ for sub-dividing all simplices yields a collection of 32 simplices, leaving the linear program with 81 decision variables and 480 constraints. The solvable $LP(\Sigma, K)$ for the CII method yields a collection of 22 simplices, leaving the linear program with 55 decision variables and 330 constraints.

■

Convergence

It is possible to prove a convergence property for the modified BBAlgorithm as follows.

Proposition 38 *For a given dynamical system, assume that Algorithm 28 terminates and produces a Lyapunov function. Then Algorithm 37 also produces a Lyapunov function.*

Proof
Since the BBAlgorithm terminates at some iteration $\hat{k}$, there exists a Bernstein basis certifiable Lyapunov function V. Let a bad simplex be a simplex not certifying the positive definiteness of V. At iteration k in the modified BBAlgorithm, the collection of simplices consists of simplices of different levels, see Definition 29. Define

$$B_k = \{\text{bad simplices in the collection at iteration k}\}, \tag{222}$$

where all these bad simplices have level $< \hat{k}$. Group the bad simplices by level as

$$\begin{aligned} B_k \mapsto i_k = (&\#\text{bad simplices of level } 0, \\ &\#\text{bad simplices of level } 1, \ldots, \\ &\#\text{bad simplices of level } \hat{k}-1) \in \mathbb{N}^{\hat{k}}. \end{aligned} \tag{223}$$

Let h_k be the minimum level of bad simplices at iteration k. Then

$$i_{k+1,j} \leq i_{k,j} \qquad \forall j \in \{0, 1, \ldots, h_k\}, \tag{224}$$

and since the CII method identifies at least one bad simplex at every iteration

$$i_{k+1} < i_k \text{ in lexicographic order on } \mathbb{N}^{\hat{k}}. \tag{225}$$

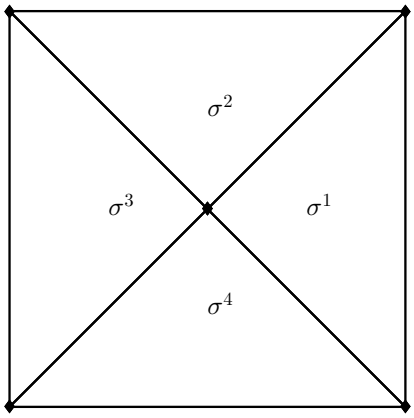
This implies the termination of the modified BBAlgorithm.

■

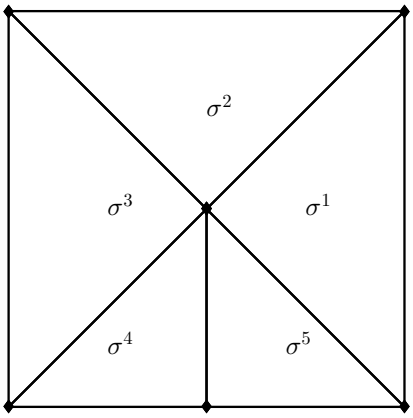
Using the CII method initially allows for the sub-division routine to create non-proper collections. This is undesirable since it disables an easy fulfilment of the continuity constraint as given in Equation (58) or Equation (60) during intermediate iterations. To understand the problem, consider Figure 27 where the initial collection $K^{\{1\}} = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$ is shown in Figure 27(a). For the sake of the argument, for some dynamical system Σ, imagine that the CII method identifies that simplex σ^4 contains constraints responsible for infeasibility of $LP(\Sigma, K^{\{1\}})$. The simplex is sub-divided by binary splitting into simplices σ^4 and σ^5 such that $K^{\{2\}} = \{\sigma^1, \ldots, \sigma^5\}$ as seen in Figure 27(b). $LP(\Sigma, K^{\{2\}})$ is also infeasible and the CII method identifies σ^5 as responsible. Another application of binary splitting yields $K^{\{3\}} = \{\sigma^1, \ldots, \sigma^6\}$ as seen in Figure 27(c). This is a non-proper collection. The sub-division of simplex σ^1 as shown in Figure 27(d) is artificially included to get a proper collection $K^{\{3^*\}} = \{\sigma^1, \ldots, \sigma^7\}$. This artificial sub-division can sometimes result in simplices which are not S-generated. This is the case in Figure 27(d) for σ^1 and σ^7. This potentially destroys the convergence result, but should σ^1 and/or σ^7 from $K^{\{3^*\}}$ be selected by the CII method in later iterations, they can easily be combined into σ^1 from $K^{\{3\}}$ again, and then sub-divided accordingly.

Using the CII method to identify simplices to sub-divide, is an attempt to obtain a solvable $LP(\Sigma, K)$ by only modifying the collection of simplices in areas of the hypercube where the constraints are not fulfilled. When working with polynomials, the fact that the coefficients on one simplex define all other coefficients (see Equation (58)) diminishes the CII method's ability to work as intended. In contrast, working with continuous piecewise-polynomials, the fact that coefficients on one simplex only define the coefficients on the facets of neighbouring simplices (see Equation (60)) allows for the CII method to work as intended.

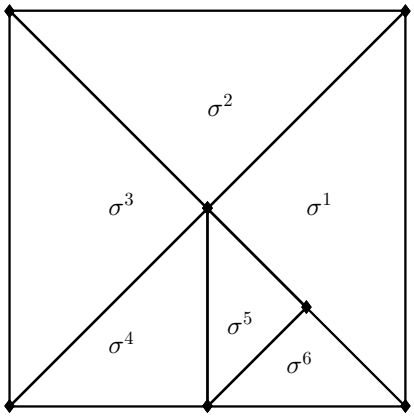
In Section 18.3, the listed results state that there is no theoretical argument for not simply using polynomial Lyapunov functions when analysing polynomial vector fields. However, the numerical and algorithmic considerations leading to the CII method is an argument for using the broader class of continuous piecewise-polynomial Lyapunov functions. I consider this realisation to be one of the main contributions of my PhD studies and I believe that further investigation of the differences between what theory deems needed and what algorithms can efficiently search for, can revolutionise software for stability certificates. It does not suffice to ask the computer to do something in one way just because theory says it is enough, we need to ask the computer to do it in an intelligent way.



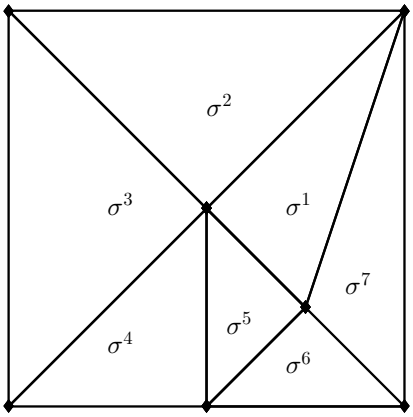
(a) Initial collection $K^{\{1\}} = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4\}$.



(b) After the first sub-division the collection is $K^{\{2\}} = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4, \sigma^5\}$.



(c) After the second sub-division the collection is $K^{\{3\}} = \{\sigma^1, \sigma^2, \sigma^3, \sigma^4, \sigma^5, \sigma^6\}$.



(d) Extra sub-division included in second sub-division to obatin proper collection.

Figure 27: The problem of possibly obtaining a non-proper collection when using the CII method. After two sub-divisions, the collection to use in the third interation $K^{\{3\}}$ is initially a non-proper collection. This is overcome by including a convention to sub-divide additional simplices to obtain proper collections, as with σ^1 in (c) to σ^1 and σ^7 in (d).

A Remark on Sparsity and Solvers

For linear programming problems, the inequality matrix sometimes possesses a certain structure, for instance block diagonal, lower triangular, or staircase [12]. Different structures offer different customised solving strategies and can potentially improve numerical stability and solver time compared to more generic strategies. When synthesising a continuous piecewise-

polynomial Lyapunov function, the coefficients in the interior of simplices are unique to their respective simplex. Because of this, it is possible to rearrange the inequality matrix into a structure known as dual block angular. This structure offers sparsity and a dedicated solver is presented in [31]. It relies on parallel implementation and requires hardware not necessarily present on an everyday laptop. It does, however, spark the hope that a customised solving strategy can help alleviate the curse of dimensionality.

21 Software

This Section combines most of the software from Part I with the developed algorithms into a single function to certify stability.

The function *analyseStability* is used to analyse the stability properties of a polynomial vector field defined by the inputs C and *gamma* on the hypercube *interval*. The function has 4 optional inputs: the degree of the Lyapunov function to search for dV; the type of Lyapunov function to search for *LyapFcn* determined as a string by either $'POL'$ for polynomial or $'CPP'$ for continuous piecewise-polynomial; the sub-division strategy *TriMeth* determined as a string by either $'ALL'$ for binary splitting on all simplices or $'CII'$ for binary splitting on the CII simplices; and whether to search for at Lyapunov function or a Lyapunov function candidate (see Chapter Instability Certification) *StbType* determined as a string by either $'STA'$ or $'INS'$. The output is a string containing the conclusion. Optional outputs are the collection of simplicies $vex, simplex$, quantities defining the synthesised Lyapunov function, and quantities defining the Lie derivative. Note that the function does not necessarily terminate.

The function *getStabilityCertificateCPPLyap* is used to synthesise a continuous piecewise-polynomial Lyapunov function for a dynamical system. It takes six inputs: C and *gamma* defining the vector field, *vex* and *simplex* defining the collection of simplices, dV is the desired degree to search for, and *StbType* is used to determine whether to search for a stability or an instability certificate. If *StbType* is unspecified, the default is to search for a stability certificate. The function uses MOSEK to solve the linear problem and returns the structure *res*, which contains the outcome of the optimisation. It also returns the number of variables n, the inequality matrix from Equation (182) *CLieTrans*, the degree of the Lie derivative *dLie*, the α-matrix for the Lie derivative *alphaLie*, the location of all grid points for the Lie derivative in the collection from Equation (27) *ctrlPointsLie*, the maps Δ^i relating the grid points for the Lie derivative to simplices in the collection as shown in Equations (52) and (53) in the matrix *simplexCtrlPointsLie*, the α-matrix for the Lyapunov function *alphaV*, the location of the grid points for the Lyapunov function *ctrlPointsV*, the maps Δ^i for the Lyapunov function *simplex-*

CtrlPointsV, and finally two vectors *CVsimplexShared* and *CVsimplexUnique* which contain indices of design variables shared by two or more simplices and indices of design variables unique to a single simplex.

The function *getLieDerivativeTrans* is used to obtain the inequality matrix from Equation (182). It takes eight inputs: the vector field being analysed defined in the Bernstein basis by coefficients *CB*, the α-matrix $alphaC$, the degree d and the maps Δ^i relating the grid points for the vector field to simplices as shown in Equations (52) and (53) *simplexCtrlPointsC*. It also takes the degree of the Lyapunov function to search for dV, the number of variables n and the collection on which everything is defined vex and $simplex$. The outputs are the inequality matrix *CLieTrans*, the α-matrix for the Lie derivative *alphaLie*, the degree of the Lie derivative *dLie*, the location of all grid points for the Lie derivative in the collection from Equation (27) *ctrlPointsLie*, the maps Δ^i relating the grid points for the Lie derivative to simplices in the collection as shown in Equations (52) and (53) in the matrix *simplexCtrlPointsLie*, the α-matrix for the Lyapunov function *alphaV*, the location of the grid points for the Lyapunov funvtion *ctrlPointsV*, and the maps Δ^i for the Lyapunov function *simplexCtrlPointsV*.

The function *getSimplexUniqueCtrlPoints* takes the maps Δ^i for the Lyapunov function *simplexCtrlPointsV* and identifies all coefficients, i.e. design variables, that are shared by two or more simplices in the collection *CVsimplexShared*, and all which are unique to a single simplex *CVsimplexUnique*.

The function *getBinarySplitting* takes the collection defined by vex and $simplex$ as inputs and performs binary splitting. An optional third input, *index*, determines which simplices to sub-divide. If left unspecified, the default is to perform binary splitting on all simplices in the collection. Should the sub-division result in a non-proper collection the function sub-divides additional simplices to obtain a proper collection. The new collection is outputted as vex and $simplex$.

The function *getStabilityCertificatePOLLyap* is used to synthesise a polynomial Lyapunov function for a dynamical system. It takes six inputs: *C* and *gamma* defining the vector field, vex and $simplex$ defining the collection of simplices, dV is the desired degree to search for, and *StbType* is used to determine whether to search for a stability or an instability certificate. If *StbType* is unspecified, the default is to search for a stability certificate. The function uses MOSEK to solve the linear problem and returns the structure *res*, which contains the outcome of the optimisation. It also returns the number of variables n, the inequality matrix from Equation (182) *CLieTrans*, the degree of the Lie derivative *dLie*, the α-matrix for the Lie derivative *alphaLie*, the location of all grid points for the Lie derivative in the collection from Equation (27) *ctrlPointsLie*, the maps Δ^i relating the grid points for the Lie derivative to simplices in the collection as shown in Equations (52) and (53) in the matrix *simplexCtrlPointsLie*, the α-matrix for the Lyapunov function *alphaV*, the

location of the grid points for the Lyapunov function *ctrlPointsV*, the maps Δ^i for the Lyapunov function *simplexCtrlPointsV*, and the matrix *B2C* which relates the coefficients on simplex σ^m to the coefficients on all simplices in the collection according to Equation (98).

Finally, the function *getLyapunovFunctionPlot* is used to visualise the synthesised Lyapunov function and Lie derivative. It creates a figure and plots the polynomial described by coefficient vector *CB*, α-matrix *alpha*, degree *d*, and on simplex *simplex*. An optional input *CtrlPointsOnPlots* enables the plotting of the control points on the graph. The default is to plot the points.

22 Design Using the Basis Polynomials

In the service of completeness, this Section briefly presents another utilisation of the Bernstein basis to synthesise Lyapunov functions. This method was devised by Sriram Sankaranarayanan (among others, see [42]) and was the reason for our collaboration and me visiting him. It does not focus on the coefficients and their sign, but instead uses a Reformulation Linearisation Technique [44].

The idea behind the reformulation linearisation technique is to introduce new variables to replace the non-linear terms in polynomial optimisation. Each different non-linear term results in one new variable. The reformulated, now linear, optimisation is then appended with constraints capturing information relating the new variables to the non-linear terms. The solution then approximates the solution to the original non-linear problem. The more information captured by the constraints, the better the approximation. For instance, if y replaces x^2 it is obvious that $y \geq 0$. If, in addition, the original problem had a bound like $-1 \leq x \leq 1$ then $0 \leq y \leq 1$ can be added to the linear problem.

Since the non-linear problem to reformulate is cast in the Bernstein basis, the non-linear terms to replace are the basis polynomials $\mathcal{B}$, see Equation (15). Chapter Bernstein Basis Properties presented properties useful for the development in this Thesis, but as mentioned in the chapter, the Bernstein basis does posses several other properties. By imposing properties like partition of unity (Equation (18)) or the non-negativity of the basis polynomials, the method in [42] creates a hierarchy of three realisations. When the search for a Lyapunov function fails using one realisation, more properties are imposed in the next.

The most resent contribution along this line of thought is [41] where the reformulation linearisation technique is paired with an iterative approach to simultaneous Lyapunov function and feedback synthesis. The authors overcome the problem of bi-linearity by fixing either the Lyapunov function or the feedback and searching for the other. This method is similar to the ap-

proach in [20] where semi-definite programming is used to synthesise Lyapunov function and feedback simultaneously, also by iteratively fixing one and searching for the other.

Instability Certification

This Chapter is concerned with the problem of possibly trying to certify the stability of a system which is in fact unstable. In this case, the linear program for synthesising a Lyapunov function will never terminate. Instead, I though of setting op the program to search for a candidate Lyapunov function with a negative semi-definite Lie derivative, but initially without any constraints on the Lyapunov function. This thought came to me during my stay with Sriram in Boulder. In [42] Sriram argued to disregard the condition on the Lyapunov function and justified it by a result from [52]. The setting was different in [42] in the way that they wanted to reduce their problem size and I wanted to be able to investigate a broader class of systems.

When a candidate Lyapunov function is found, a sub-sequent determination of the sign of the candidate Lyapunov function then determines if the function is a Lyapunov function which certifies stability, or if the function certifies instability (or if the function is sign indefinite and no conclusion can be drawn). This enables the automated analysis of systems which may or may not be stable, and the algorithm can accommodate both cases. Seeing as the positivity certification of a known polynomial plays a role, the considerations in Part I suddenly become interesting again.

23 Modifying the Linear Program

In terms of setting up the linear program, the procedure is exactly like before, except that the lower bounds on the design variables are disregarded. This leaves us with the program

$$\begin{aligned} &\min_{CV} \\ &\text{s.t.} \quad l^c \leq A\, CV^T \leq u^c, \end{aligned} \tag{226}$$

where A is the matrix from Equation (181) or Equation (185) depending on whether a polynomial or continuous piecewise-polynomial candidate Lyapunov function is synthesised. Contrary to the previous chapter, no objective function is considered in this Chapter.

The linear program for synthesising a candidate Lyapunov function will be referred to as $LP_c(\Sigma, K)$. When $LP_c(\Sigma, K)$ is infeasible, the collection of simplices is modified according to the procedures introduced for the BBAlgorithm and modified BBAlgorithm. Initially, I expected the runtime for the analysis to be at least equal to that of $LP(\Sigma, K)$, and possibly way faster, since the problem is less constrained. However, when testing the technique on examples, the runtime when searching for a candidate Lyapunov function turned out to be almost equal to searching for a Lyapunov function. In addition, the sub-sequent sign determination turned out to be very expensive, as reported in the examples below. Regardless, remember that only searching for a candidate allows for a program which terminates even if the system under investigation is unstable. This is a valuable feature.

24 Unstable Systems

This Section considers two systems, which are both analysed by searching for a Lyapunov function and for a candidate Lyapunov function. They are investigated under both stable and unstable stability properties. A stable system is made unstable by simply negating the sign of all coefficients in the coefficient matrix c.

Simplex Example

This Example considers a vector field $f : \mathbb{R}^2 \to \mathbb{R}^2$, taken from [42], given as

$$\begin{aligned} c &= \begin{bmatrix} -1.5 & 0 & -1 & 0.5 & 0.5 & -2 & 1 \\ 0 & -0.5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \gamma &= \begin{bmatrix} 1 & 0 & 2 & 1 & 0 & 3 & 2 \\ 0 & 1 & 0 & 1 & 2 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{227}$$

when analysing both the stable and unstable version, using both polynomial (POL) and continuous piecewise-polynomial (CPP) (candidate) Lyapunov functions. The analysis is done using the modified BBAlgorithm with the degree input set to $d_V = 2$. The findings are reported in Table 1, which is read as follows. On the left, the actual stability property of the system is listed. Then the settings of the algorithm are seen. Next, the runtimes are reported, where Solve LP refers to the time needed to solve the linear program and Cert Pos is the time needed for subsequent sign determination when synthesising candidate Lyapunov functions. The times are in seconds and the * indicates the analysis shown in Figure 28.

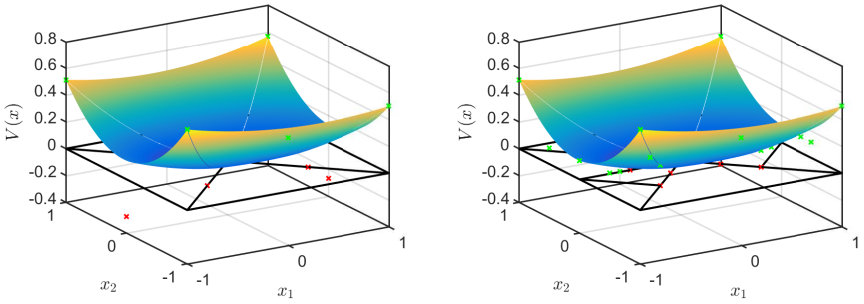
When analysing the stable system using a continuous piecewise-polynomial (candidate) Lyapunov function, both $LP(\Sigma, K)$ and $LP_c(\Sigma, K)$ are solvable with the initial collection consisting of 4 simplices. Certifying the posi-

		CPP		POL	
	Setting	Solve LP	Cert Pos	Solve LP	Cert Pos
Stable	'STA'	0.57 s	N/A	2.1 s	N/A
	'INS'	0.60 s*	0.38 s*	1.2 s	0 s
Unstable	'STA'	∞	N/A	∞	N/A
	'INS'	0.53 s	0.31 s	1.1 s	0 s

Table 1: Runtimes for stability and instability analysis of (227) using both polynomial and continuous piecewise-polynomial (candidate) Lyapunov functions. The analysis is done with the modified BBAlgorithm and a degree of 2. The * indicates the analysis shown in Figure 28.

tivity of the candidate Lyapunov function requires sub-division, resulting in a collection of 8 simplices. This situation is shown in Figure 28. Using a polynomial (candidate) Lyapunov function requires one sub-division of the initial collection before $LP(\Sigma, K)$ and $LP_c(\Sigma, K)$ become solvable. In this case, certifying the positivity of the candidate Lyapunov function does not require any sub-division since all coefficients of the candidate Lyapunov function were automatically non-negative. As such, the solution to $LP_c(\Sigma, K)$ automatically fulfilled the positivity criteria of $LP(\Sigma, K)$ without it being enforced.

When analysing the unstable system, the program searching for a Lyapunov function can never terminate. When searching for a continuous piecewise-polynomial candidate Lyapunov function, $LP_c(\Sigma, K)$ is solvable without



(a) Continuous piecewise-polynomial candidate Lyapunov function for (227).

(b) Candidate Lyapunov function certified non-negative.

Figure 28: Comparison of the solution to stable (227) using continuous piecewise-polynomial candidate Lyapunov function before and after certifying the non-negativity. Note on Figure (b) that the collection is not a proper collection of simplices. This does not matter in this setting, since the continuity of the polynomial is embedded in $LP_c(\Sigma, K)$ and the certification process treats each initial simplex individually. Each sub-division will result in proper collections of simplices being created from the individual initial simplices. The red coefficients on Figure (b) are in the magnitude of 10^{-11}.

	Setting	Solve LP	Cert Pos
Stable	'STA'	~3 min	N/A
	'INS'	~3 min	~137 min
Unstable	'STA'	∞	N/A
	'INS'	~3 min	~99 min

Table 2: Runtimes for stability and instability analysis of the comprehensive example using continuous piecewise-polynomial (candidate) Lyapunov functions.

any sub-division of the initial triangulation, and the certification of the candidate Lyapunov function requires sub-dividing from 4 to 8 simplices. Considering a polynomial candidate Lyapunov function $LP_c(\Sigma, K)$ requires one sub-division before it becomes solvable. As with the stable system, certifying the candidate Lyapunov function does not require any sub-division, since the coefficients were all automatically non-negative.

■

Comprehensive Example

This Example considers a 4 dimensional system of degree 6 taken from [42]. The vector field has a total of 161 different monomial combinations and is not printed in the thesis. The system is given in the function *sampleSystems*, see the software section below. The system was analysed, both as stable and unstable, searching both for a candidate Lyapunov function and for a Lyapunov function. In both cases, only continuous piecewise-polynomials were sought synthesised and the degree was $d_V = 6$. The results are reported in Table 2, which is read as follows.

On the left, the actual stability property of the system is listed, followed by the setting in the algorithm. Solve LP contains the information regarding runtime reported in minutes. Cert Pos indicates the runtime in minutes for certifying the positivity of the candidate Lyapunov function in the stable case, or the runtime for certifying the negativity of the candidate Lyapunov function in the unstable case.

Observe that the computations needed to certify the stability of a stable system now take significantly longer than they did before allowing for analysis of unstable systems. The advantage, of course, is the added flexibility seen in that the analysis of an unstable system can never terminate when trying to certify stability. In this case, the long runtime is simply a necessary evil, in order to ensure termination of the program.

Note that in the three solvable cases, $LP(\Sigma, K)$ and $LP_c(\Sigma, K)$ are solvable using the initial collection K_0 consisting of 48 simplices as obtained from Kuhn's triangulation of the 4 dimensional unit cube, see Definition 3. Since no sub-division is needed, the BBAlgorithm and the modified BBAlgorithm

are equal. However, in order to certify the positivity in the stable case, the 48 simplices are sub-divided into 388 simplices, where each of the original 48 simplices are sub-divided into between 1 and 23 simplices, before obtaining the certificate of positivity. In the unstable case, the 48 simplices are sub-divided into 329 simplices, where each of the original 48 simplices are sub-divided into between 1 and 24 simplices, before obtaining the certificate of negativity.

■

25 Software

The modifications needed in order to accommodate the analysis of unstable systems is straightforward, and are in fact already present in the software presented in Section 21. The parameter *StbType* works as an input to *analyseStability* and is passed to *getStabilityCertificateCPPLyap* and *getStabilityCertificatePOLLyap*. If set to allow analysis of unstable systems, the lower bounds on the design variables are set to $-\infty$ and the objective function is set to zero. The objective function is set to zero to ensure that the linear program does not become dual infeasible due to an unbounded primal objective function.

The function *determineSignOfLyapunovFunction* can be used to determine the sign of any polynomial defined on a collection of simplices. It analyses the coefficients at the vertices in the collection. Based on them being positive or negative, *determineSignOfLyapunovFunction* tries to certify the non-negativity or non-positivity of the polynomial. If the sign of the vertex coefficients are mixed, the conclusion is that the polynomial is sign indefinite. The function takes seven inputs. C, $simplexCtrlPoints$, and $alpha$ are the coefficients, the maps Δ^i, and the α-matrix respectively. vex and $simplex$ define the collection and n and d are the number of variables and the degree. The function outputs *res*, which is a string containing the conclusion of the analysis and CC, $alphaC$, $vexC$ and $simplexC$ which are the coefficients and α-matrix on the resulting collection.

The function *getVertexCoefficientsComplex* is used to single out coefficients located at the vertices in a collection. The inputs are the coefficient vector C, the maps Δ^i in $simplexCtrlPoints$, and the α-matrix in $alpha$. The output $VexCoef$ contains the coefficients at the vertices.

The *.m*-script *sampleSystems* contains a collection of polynomial vector fields, which are stable. The comprehensive example from above is case 15. All systems analysed in the thesis can be found in the script.

References

[1] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A. Parrilo. "A Globally Asymptotically Stable Polynomial Vector Field with no Polynomial Lyapunov Function". In: *Proceedings of the Conference on Decision and Control* (2011), pp. 7579–7580.

[2] Amir Ali Ahmadi and Pablo A. Parrilo. "Converse Results on Existence of Sum of Squares Lyapunov Functions". In: *Proceedings of the Conference on Decision and Control* (2011), pp. 6516–6521.

[3] Erling D. Andersen. "Certificates of Primal or Dual Infeasibility in Linear Programming". In: *Computational Optimizations and Applications* (2001), pp. 171–183.

[4] Erling D. Andersen. *How to Use Farkas' Lemma to Say Something Important About Infeasible Linear Problems*. Tech. rep. MOSEK, 2013. URL: `http://docs.mosek.com/whitepapers/infeas.pdf`.

[5] Jasbir S. Arora. *Introduction to Optimum Design*. 3. Elsevier Inc., 2012.

[6] Andrea Bacciotti and Lionel Rosier. *Liapunov Functions and Stability in Control Theory*. Springer, 2005.

[7] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Vol. 10. Springer, 2006.

[8] J. Berchtold and A. Bowyer. "Robust Artihmetic for Multivariate Bernstein-Form Polynomials". In: *Computer-Aided Design* 32 (2000), pp. 681–689.

[9] Carl de Boor. "B-Form Basics". In: *Geometric Modeling* (1987), pp. 131–148.

[10] Fatima Boudaoud, Fabrizio Caruso, and Marie-Françoise Roy. "Certificates of Positivity in the Bernstein Basis". In: *Discrete and Computational Geometry* 39 (2007), pp. 639–655.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[12] Stephen P. Bradley, Arnoldo C. Hax, and Thomas L. Magnanti. *Applied Mathematical Programming*. Addison-Wesley Publishing Company, 1977.

[13] Gerald Farin. "Triangular Bernstein-Bézier Patches". In: *Computer Aided Geometric Design* 3.2 (1986), pp. 83–127.

[14] Rida T. Farouki. *The Bernstein Polynomial Basis: A Centennial Retrospective*. Tech. rep. Department of Mechanical and Aerospace Engineering, University of California, 2012.

[15] Rida T. Farouki and V. T. Rajan. "Algorithms for Polynomails in Bernstein Form". In: *Computer Aided Geometric Design* 5 (1988), pp. 1–26.

[16] Peter A. Giesl and Sigurdur F. Hafstein. "Revised CPA Method to Compute Lyapunov Functions for Nonlinear Systems". In: *Journal of Mathematical Analysis and Applications* 410.1 (2014), pp. 292–306.

[17] Tim Goodman and Jörg Peters. "Bézier Nets, Convexity and Subdivision on Higher Dimensional Simplices". In: *Computer Aided Geometric Design* 12 (1994), pp. 53–65.

[18] Wolfgang Hahn. *Stability of Motion*. Springer-Verlag New York Inc., 1967.

[19] Didier Henrion, Jean-Bernard Lasserre, and Johan Löfberg. "GloptiPoly 3: Moments, Optimization and Semidefinite Programming." In: *Optimization Methods and Software* 24 (2009), pp. 761–779.

[20] Zachary William Jarvis-Wolszek. "Lyapunov Based Analysis and Controller Synthesis for Polynomial Systems using Sum-of-Squares Optimization". PhD thesis. University of California, Berkeley, 2003.

[21] Reza Kamyar, Chaitanya Murti, and Matthew M. Peet. "Constructing Piecewise-Polynomial Lyapunov Functions for Local Stability of Nonlinear Systems Using Handelman's Theorem". In: *Proceedings of the Conference on Decision and Control* (2014), pp. 5481–5487.

[22] Christopher M. Kellett. "A Compendium of Comparison Function Results". In: *Mathematics of Control, Signals, and Systems* 26.3 (2014), pp. 339–374.

[23] Hassan K. Khalil. *Nonlinear Systems*. Vol. 3. Prentice Hall, 1996.

[24] V. Lakshmikantam and A. A. Martynyuk. "Lyapunov's Direct Method in Stability Theory (Review)". In: *Prikladnaya Mekhanika* 28.3 (1992). Translation, pp. 135–144.

[25] Jean Bernard Lasserre. *Moments, Positive Polynomials and Their Applications*. Ed. by Jean Bernard Lasserre. Vol. 1. Imperial College Press Optimization Series. Imperial College Press, 2010.

[26] Richard Leroy. "Certificates of Positivity in the Simplicial Bernstein Basis". In: *Archives: hal-00589945* (2011).

[27] Tobias Leth, Christoffer Sloth, and Rafał Wisniewski. “Lyapunov Function Synthesis - Algorithm and Software”. In: *Proceedings of the Multi-Conference on Systems and Control* (2016), pp. 641–647.

[28] Tobias Leth, Rafał Wisniewski, and Christoffer Sloth. “On the Existence of Polynomial Lyapunov Funcitons for Rationally Stable Vector Fields”. In: *Proceedings of the Conference on Decision and Control* (2017).

[29] Tobias Leth, Christoffer Sloth, Rafał Wisniewski, and Sriram Sankaranarayanan. “Lyapunov Functions Synthesis - Infeasibility and Farkas’ Lemma”. In: *Proceedings of the IFAC World Congress* (2017), pp. 1703–1708.

[30] Tobias Leth, Carsten Skovmose Kallesøe, Christoffer Sloth, and Rafał Wisniewski. “Stability of Drinking Water Distribution Network”. In: *Proceedings of the European Control Conference* (2016), pp. 1764–1769.

[31] Miles Lubin, J. A. Julian Hall, Cosmin G. Petra, and Mihai Anitescu. “Parallel Distributed-Memory Simplex for Large-Scale Stochastic LP Problems”. In: *Computational Optimizations and Applications* 55.3 (2013), pp. 571–596.

[32] Esmeralda Mainar and J.M. Peña. “Evaluation Algorithms for Multivariate Polynomails in Bernstein-Bézier Form”. In: *Journal of Approximation Theory* 143.1 (2006), pp. 44–61.

[33] Sigurdur F. Marinòsson. “Stability Analysis of Nonlinear Systems with Linear Programming - A Lyapunov Functions Based Approach”. PhD thesis. Gerhard-Mercator-University, 2002.

[34] Jose L. Massera. “Contributions to Stability Theory”. In: *Annals of Mathematics* 64.1 (1956), pp. 182–206.

[35] César Muñoz and Anthony Narkawicz. “Formalization of Bernstein Polynomials and Applications to Global Optimization”. In: *Journal of Automated Reasoning* 51.2 (2013), pp. 151–196.

[36] P. S. V. Nataraj and M. Arounassalame. “A New Subdivision Algorithm for the Bernstein Polynomial Approach to Global Optimization”. In: *International Journal of Automation and Computing* 4.4 (2007), pp. 342–352.

[37] Pablo A. Parrilo. “Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization”. PhD thesis. California Institute of Technology, 2000.

[38] Matthew M. Peet. “Exponentially Stable Nonlinear Systems Have Polynomial Lyapunov Functions on Bounded Regions”. In: *IEEE Transactions on Automatic Control* 54.5 (2009), pp. 979–987.

[39] Victoria Powers. “Positive Polynomial and Sums of Squares: Theory and Practice”. In: *Real Algebraic Geometry* (2011), pp. 1–28.

[40] Stephen Prajna, Antonis Papachristodoulou, Peter Seiler, and Pablo A. Parrilo. "SOSTOOLS and its Control Applications". In: *Positive Polynomials in Control* 312 (2005), pp. 273–292.

[41] Mohamed Amin Ben Sassi, Ezio Bartocci, and Sriram Sankaranarayanan. "A Linear Programming-Based Iterative Approach to Stabilizing Polynomial Dynamics". In: *Proceedings of the IFAC World Congress* (2017), pp. 10951–10958.

[42] Mohamed Amin Ben Sassi, Sriram Sankaranarayanan, Xin Chen, and Erika Ábrahám. "Linear Relaxations of Polynomial Positivity for Polynomial Lyapunov Function Synthesis". In: *Journal of Mathematical Control and Information* (2015), pp. 1–34.

[43] Konrad Schmudgen. "Around Hilbert's 17TH Problem". In: *ISMP* Extra (2012), pp. 433–438.

[44] Hanif D. Sherali and Cihan H. Tuncbilek. "A Global Optimization Algorithm for Polynomial Programming Problems Using a Reformulation-Linearization Technique". In: *Journal of Global Optimization* 2 (1992), pp. 101–112.

[45] Christoffer Sloth. "Nonnegative Polynomial with no Certificate of Nonnegativity in the Simplicial Bernstein Basis". In: *Archives* (2017).

[46] Christoffer Sloth and Rafał Wisniewski. "Robust Stability of Switched Systems". In: *Proceedings of the Conference on Decision and Control* (2014), pp. 4685–4690.

[47] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[48] Georgi V. Smirnov. *Introduction to the Theory of Differential Inclusions*. Vol. 41. Graduate Studies in Mathematics. American Mathematical Society, 2002.

[49] Gilbert Stengle. "A Nullstellensatz and a Positivstellensatz in Semialgebraic Geometry". In: *Mathematische Annalen* 207 (1974), pp. 87–97.

[50] Romain Testylier and Thao Dang. "Analysis of Parametric Biological Models with Non-Linear Dynamics". In: *Proceedings of the Workshop on Hybrid Systems and Biology* (2012).

[51] Jihad Titi and Jürgen Garloff. "Matrix Methods for the Simplicial Bernstein Representation and for the Evaluation of Multivariate Polynomials". In: *Konstanzer Schriften in Mathematik* (2017).

[52] A. Vannelli and M. Vidyasagar. "Maximal Lyapunov Functions and Domains of Attraction for Autonomous Nonlinear Systems". In: *Automatica* 21.1 (1985), pp. 69–80.

Appendix A

Selected Basis Polynomials

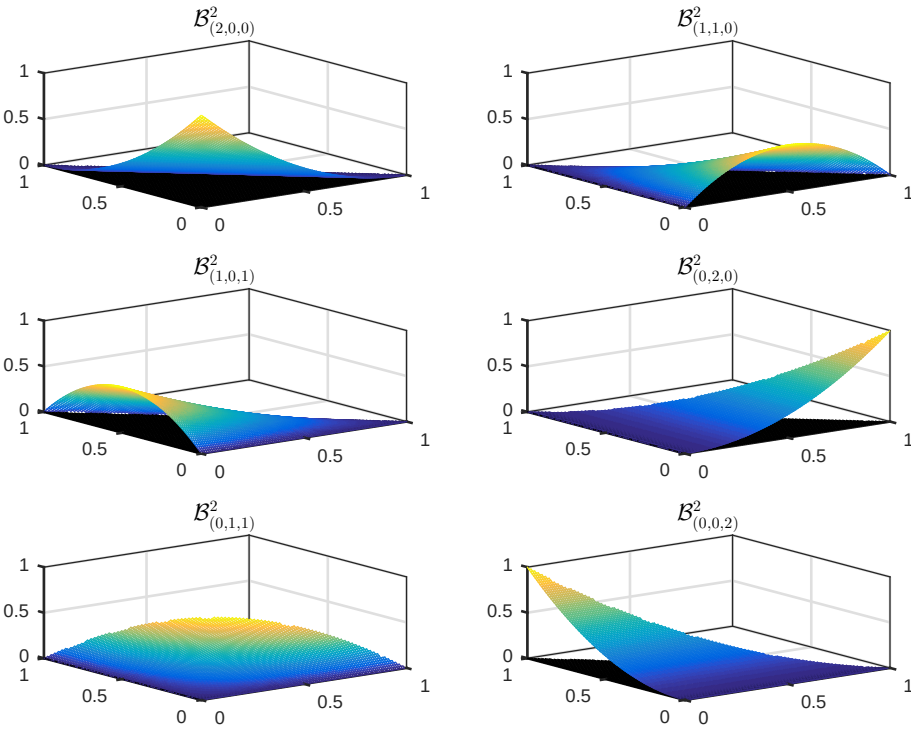


Figure A.1: Bernstein basis polynomials in two dimension of degree 2. The shaded area is the simplex.

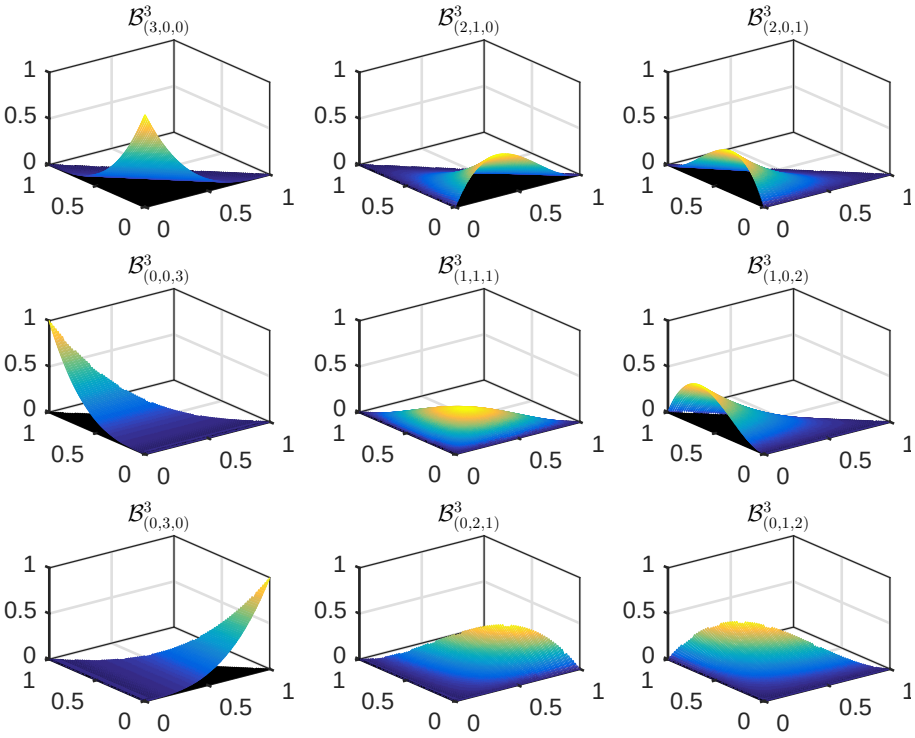


Figure A.2: Bernstein basis polynomials in two dimension of degree 3 with $\mathcal{B}^3_{(1,2,0)}$ omitted. See Figure 7 for $\mathcal{B}^3_{(1,2,0)}$. The shaded area is the simplex.

Appendix B

Basis Transformation Derivation

Monomial to Bernstein

This Section considers how to obtain the matrix A as defined in Equation (29). The matrix is used to transform descriptions of a polynomial in the monomial basis to a description in the Bernstein basis on a given simplex. The derivation is eventually limited to an arbitrary two dimensional simplex but readily available for algorithmic implementation for arbitrary dimension and simplex. This is similar to [32] where the derivation was limited to three dimensions but restricted to the standard simplex.

Polynomials in the monomial basis were described by Equation (11), repeated here for convenience, as

$$p(x) = \sum_{k=1}^{K} c_k x^{\gamma_k} \tag{B.1}$$

where $x \in \mathbb{R}^n$, $c = [c_1 \; c_2 \; \cdots \; c_K]$ is the coefficient vector, and

$$\gamma = \begin{bmatrix} \gamma_1(1) & \gamma_2(1) & \cdots & \gamma_K(1) \\ \gamma_1(2) & \gamma_2(2) & \cdots & \gamma_K(2) \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_1(n) & \gamma_2(n) & \cdots & \gamma_K(n) \end{bmatrix} \tag{B.2}$$

$$x^{\gamma_k} = x_1^{\gamma_k(1)} x_2^{\gamma_k(2)} \cdots x_n^{\gamma_k(n)}. \tag{B.3}$$

The degree of p is

$$d = \max_{k \in \{1,\ldots,K\}} \sum_{i=1}^{n} \gamma_k(i). \tag{B.4}$$

That is, the maximal degree of any monomial is equal to the actual degree of p.

Since

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \sigma_1(1) & \cdots & \sigma_n(1) \\ \sigma_0(2) & \sigma_1(2) & \cdots & \sigma_n(2) \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_0(n) & \sigma_1(n) & \cdots & \sigma_n(n) \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \cdots \\ \lambda_n \end{bmatrix}, \tag{B.5}$$

from Equation (4), expanding the expressing for p described in the monomial basis using the multinomial theorem (see. Equation (18)) yields

$$p = \sum_{k=1}^{K} c_k \prod_{i=1}^{n} x_i^{\gamma_k(i)} \tag{B.6}$$

$$= \sum_{k=1}^{K} c_k \prod_{i=1}^{n} \left([\sigma_0(i) \ \cdots \ \sigma_n(i)] \begin{bmatrix} \lambda_0 \\ \cdots \\ \lambda_n \end{bmatrix} \right)^{\gamma_k(i)} \tag{B.7}$$

$$= \sum_{k=1}^{K} c_k \prod_{i=1}^{n} \sum_{|\beta|=\gamma_k(i)} \binom{\gamma_k(i)}{\beta} \prod_{t=0}^{n} (\sigma_t(i)\lambda_t)^{\beta_t} \tag{B.8}$$

$$= \sum_{k=1}^{K} c_k \prod_{i=1}^{n} \sum_{|\beta|=\gamma_k(i)} \underbrace{\prod_{t=0}^{n} \sigma_t(i)^{\beta_t}}_{b(i)_\beta} \underbrace{\binom{\gamma_k(i)}{\beta} \prod_{t=0}^{n} \lambda_t^{\beta_t}}_{\mathcal{B}_\beta^{\gamma_k(i)}} \tag{B.9}$$

$$= \sum_{k=1}^{K} c_k \prod_{i=1}^{n} \sum_{|\beta|=\gamma_k(i)} b(i)_\beta \mathcal{B}_\beta^{\gamma_k(i)} \tag{B.10}$$

Note that the products over sums of β are products of Bernstein polynomials of possibly different degrees, $\gamma_k(i)$. The multiplication over i is done using Equation (114). For $n = 2$ this gives

$$p = \sum_{k=1}^{K} c_k \sum_{|\kappa|=\gamma_k(1)+\gamma_k(2)} \sum_{\substack{|\kappa-\beta_1|=\gamma_k(2) \\ \kappa-\beta_1 \geq 0}} b(1)_{\beta_1} b(2)_{\kappa-\beta_1} \frac{\binom{\gamma_k(1)}{\beta_1}\binom{\gamma_k(2)}{\kappa-\beta_1}}{\binom{\gamma_k(1)+\gamma_k(2)}{\kappa}} \mathcal{B}_\kappa^{\gamma_k(1)+\gamma_k(2)} \tag{B.11}$$

where β_1 and $\beta_2 = \kappa - \beta_1$ comes from the multiplication.
Note now that the sum over k is a summation of Bernstein polynomials of possibly different degrees $(\gamma_1(1) + \gamma_1(2), \gamma_2(1) + \gamma_2(2), \ldots, \gamma_K(1) + \gamma_K(2))$.

To add polynomials in the Bernstein basis they must be described in a basis of the same degree. Since the degree of p was

$$d = \max_{k \in \{1,\ldots,K\}} \sum_{i=1}^{n} \gamma_k(i), \tag{B.12}$$

at least one of the Bernstein polynomials in the summation is of degree d and the rest will all be of equal or lower degree. By degree elevation, see Equation (116), a relation between Bernstein basis polynomials of different degrees d and $\check{d}$, with $d > \check{d}$, is

$$\mathcal{B}_{\beta}^{\check{d}} = \sum_{|\alpha|=d} \frac{\binom{\check{d}}{\beta}\binom{d-\check{d}}{\alpha-\beta}}{\binom{d}{\alpha}} \mathcal{B}_{\alpha}^{d}. \tag{B.13}$$

Using this the monomial description is transformed to Bernstein description of degree d.

$$p = \sum_{k=1}^{K} c_k \sum_{|\kappa|=\gamma_k(1)+\gamma_k(2)} \sum_{\substack{|\kappa-\beta_1|=\gamma_k(2) \\ \kappa-\beta_1 \geq 0}} b(1)_{\beta_1} b(2)_{\kappa-\beta_1} \frac{\binom{\gamma_k(1)}{\beta_1}\binom{\gamma_k(2)}{\kappa-\beta_1}}{\binom{\gamma_k(1)+\gamma_k(2)}{\kappa}} \sum_{|\alpha|=d} \frac{\binom{\gamma_k(1)+\gamma_k(2)}{\kappa}\binom{d-(\gamma_k(1)+\gamma_k(2))}{\alpha-\kappa}}{\binom{d}{\alpha}} \mathcal{B}_{\alpha}^{d} \tag{B.14}$$

$$= \sum_{|\alpha|=d} \sum_{k=1}^{K} c_k \sum_{|\kappa|=\gamma_k(1)+\gamma_k(2)} \sum_{\substack{|\kappa-\beta_1|=\gamma_k(2) \\ \kappa-\beta_1 \geq 0}} b(1)_{\beta_1} b(2)_{\kappa-\beta_1} \frac{\binom{\gamma_k(1)}{\beta_1}\binom{\gamma_k(2)}{\kappa-\beta_1}}{\binom{\gamma_k(1)+\gamma_k(2)}{\kappa}} \frac{\binom{\gamma_k(1)+\gamma_k(2)}{\kappa}\binom{d-(\gamma_k(1)+\gamma_k(2))}{\alpha-\kappa}}{\binom{d}{\alpha}} \mathcal{B}_{\alpha}^{d} \tag{B.15}$$

$$= \sum_{|\alpha|=d} \sum_{k=1}^{K} c_k \sum_{|\kappa|=\gamma_k(1)+\gamma_k(2)} \sum_{\substack{|\kappa-\beta_1|=\gamma_k(2) \\ \kappa-\beta_1 \geq 0}} b(1)_{\beta_1} b(2)_{\kappa-\beta_1} \frac{\binom{\gamma_k(1)}{\beta_1}\binom{\gamma_k(2)}{\kappa-\beta_1}\binom{d-(\gamma_k(1)+\gamma_k(2))}{\alpha-\kappa}}{\binom{d}{\alpha}} \mathcal{B}_\alpha^d \tag{B.16}$$

Comparing (26) and (B.16) the following relation is obvious

$$b_\alpha(p,d,\sigma) = \sum_{k=1}^{K} c_k \sum_{|\kappa|=\gamma_k(1)+\gamma_k(2)} \sum_{\substack{|\kappa-\beta_1|=\gamma_k(2) \\ \kappa-\beta_1 \geq 0}} b(1)_{\beta_1} b(2)_{\kappa-\beta_1} \frac{\binom{\gamma_k(1)}{\beta_1}\binom{\gamma_k(2)}{\kappa-\beta_1}\binom{d-(\gamma_k(1)+\gamma_k(2))}{\alpha-\kappa}}{\binom{d}{\alpha}} \tag{B.17}$$

$$\forall\ |\alpha| = d.$$

Equation (B.17) was derived for $n = 2$, but the expansion needed from Equation (B.10) to Equation (B.11) can sequentially be extended to arbitrary n. The operations are implemented in loops and no further expansion is needed. The function *getMon2BernSimplex* implements the above as described in Section 4.

Bernstein to Bernstein

This Section considers how to obtain the matrix ${}^iB_j$ as defined in Equation (32). The matrix is used to transform descriptions of a polynomial on simplex σ^j to simplex σ^i. The generic method of collecting terms and equating to zero can be applied to any two bases. However, limiting the operation offers computational advantages. The following procedure is for the scenario where the two simplices overlaps with one facet, e.g. there is only one vertex unique to either simplex. This is similar to the "Domain Transformation" presented in [13], although the derivation presented here follows that of [45]. Applying the procedure $n+1$ times enables the transformation of all vertices, one by one, thus eliminating the limitation. This restriction is the argument behind the simplex numbering convention introduced in Section 3.

Figure B.1 illustrates the situation. A polynomial p is described in the Bernstein basis of degree D on σ, and it is desired to describe it on $\tilde{\sigma}$ instead, e.g.

$$p = \sum_{|\alpha|=D} b_\alpha(p,D,\sigma)\mathcal{B}_\alpha^D(\sigma) = \sum_{|\gamma|=D} b_\gamma(p,D,\tilde{\sigma})\mathcal{B}_\gamma^D(\tilde{\sigma}), \tag{B.18}$$

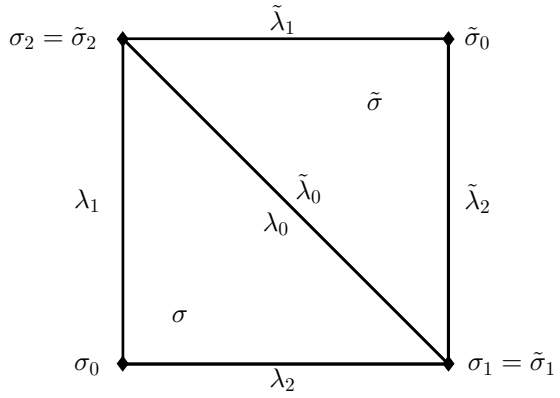


Figure B.1: Simplex σ transformed into simplex $\tilde{\sigma}$. Since only one vertex is changed the transformation is computationally efficient.

where $b(p, D, \sigma)$ is known and $b_\gamma(p, D, \tilde{\sigma})$ is unknown. To derive the relation between b_α and b_γ, first recall from the definition of barycentric coordinates that

$$x = \lambda_0\sigma_0 + \lambda_1\sigma_1 + \lambda_2\sigma_2 = \tilde{\lambda}_0\tilde{\sigma}_0 + \tilde{\lambda}_1\tilde{\sigma}_1 + \tilde{\lambda}_2\tilde{\sigma}_2. \tag{B.19}$$

Only one vertex is unique to $\tilde{\sigma}$, and it can be described as a linear combination of the original vertices by

$$\tilde{\sigma}_0 = \beta_0\sigma_0 + \beta_1\sigma_1 + \beta_2\sigma_2 \tag{B.20}$$

with $\beta_0 + \beta_1 + \beta_2 = 1$ and, to avoid $\tilde{\sigma}$ being degenerate, $\beta_0 \neq 0$. For unknown β_i's they are determined by

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sigma_0(1) & \sigma_1(1) & \sigma_2(1) \\ \sigma_0(2) & \sigma_1(2) & \sigma_2(2) \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\sigma}_0(1) \\ \tilde{\sigma}_0(2) \\ 1 \end{bmatrix}. \tag{B.21}$$

Substituting $\tilde{\sigma}_i$ with σ_i yields

$$x = \tilde{\lambda}_0(\beta_0\sigma_0 + \beta_1\sigma_1 + \beta_2\sigma_2) + \tilde{\lambda}_1\sigma_1 + \tilde{\lambda}_2\sigma_2 \tag{B.22}$$

$$= (\tilde{\lambda}_0\beta_0)\sigma_0 + (\tilde{\lambda}_0\beta_1 + \tilde{\lambda}_1)\sigma_1 + (\tilde{\lambda}_0\beta_2 + \tilde{\lambda}_2)\sigma_2, \tag{B.23}$$

which identifies

$$\lambda_0 = \beta_0\tilde{\lambda}_0, \qquad \lambda_1 = \beta_1\tilde{\lambda}_0 + \tilde{\lambda}_1, \qquad \lambda_2 = \beta_2\tilde{\lambda}_0 + \tilde{\lambda}_2. \tag{B.24}$$

To obtain the relationship between the basis polynomials on the two simplices, first consider the relationship between the barycentric coordinates raised to a given multi-index α as

$$\lambda^\alpha = (\beta_0\tilde{\lambda}_0)^{\alpha_0}(\beta_1\tilde{\lambda}_0 + \tilde{\lambda}_1)^{\alpha_1}(\beta_2\tilde{\lambda}_0 + \tilde{\lambda}_2)^{\alpha_2}, \tag{B.25}$$

and expand using the multinomial theorem (see. Equation (18))

$$\lambda^{\alpha} = \beta_0^{\alpha_0}\tilde{\lambda}_0^{\alpha_0}\left(\sum_{k=0}^{\alpha_1}\binom{\alpha_1}{k}\beta_1^{k}\tilde{\lambda}_0^{k}\tilde{\lambda}_1^{\alpha_1-k}\right)\left(\sum_{j=0}^{\alpha_2}\binom{\alpha_2}{j}\beta_2^{j}\tilde{\lambda}_0^{j}\tilde{\lambda}_2^{\alpha_2-j}\right). \tag{B.26}$$

Expanding the sums, and noting that $\alpha_0 + k + j = \gamma_0$ enables the following relation

$$\lambda^{\alpha} = \sum_{\substack{|\gamma|=D \\ \gamma_1\leq\alpha_1 \\ \gamma_2\leq\alpha_2}}\binom{\alpha_1}{\alpha_1-\gamma_1}\binom{\alpha_2}{\alpha_2-\gamma_2}\beta_0^{\alpha_0}\beta_1^{\alpha_1-\gamma_1}\beta_2^{\alpha_2-\gamma_2}\tilde{\lambda}^{\gamma}. \tag{B.27}$$

Scaling the barycentric coordinates according to Equation (15) gives

$$\mathcal{B}_{\alpha}^{D}(\sigma) = \sum_{\substack{|\gamma|=D \\ \gamma_1\leq\alpha_1 \\ \gamma_2\leq\alpha_2}}\binom{\alpha_1}{\alpha_1-\gamma_1}\binom{\alpha_2}{\alpha_2-\gamma_2}\frac{\binom{D}{\alpha}}{\binom{D}{\gamma}}\beta_0^{\alpha_0}\beta_1^{\alpha_1-\gamma_1}\beta_2^{\alpha_2-\gamma_2}\mathcal{B}_{\gamma}^{D}(\tilde{\sigma}) \tag{B.28}$$

$$= \sum_{\substack{|\gamma|=D \\ \gamma_1\leq\alpha_1 \\ \gamma_2\leq\alpha_2}}\frac{\gamma_0!}{\alpha_0!(\alpha_1-\gamma_1)!(\alpha_2-\gamma_2)!}\beta_0^{\alpha_0}\beta_1^{\alpha_1-\gamma_1}\beta_2^{\alpha_2-\gamma_2}\mathcal{B}_{\gamma}^{D}(\tilde{\sigma}). \tag{B.29}$$

Substituting this into Equation (B.18) gives the relation

$$b_{\gamma}(p,D,\tilde{\sigma}) = \sum_{\substack{|\alpha|=D \\ \alpha_1\geq\gamma_1 \\ \alpha_2\geq\gamma_2}}\frac{\gamma_0!}{\alpha_0!(\alpha_1-\gamma_1)!(\alpha_2-\gamma_2)!}\beta_0^{\alpha_0}\beta_1^{\alpha_1-\gamma_1}\beta_2^{\alpha_2-\gamma_2}b_{\alpha}(p,D,\sigma), \tag{B.30}$$

which, by arranging the entries according to vertex numbering yields ${}^{i}B_j$.

The derivation above extends naturally to $n \geq 3$ as long there is only one unique vertex in the simplices.

The function *getCtrlPointBernTrans* implements the above as described in Section 4, where the description on one simplex can be transformed to descriptions on several simplices directly.

Appendix C

Positivity Certification By Dimension Elevation

To derive the equivalence between sub-division and dimension elevation, first consider the procedure of dimension elevation of a polynomial in the monomial basis. A polynomial in the monomial basis was defined by Equation (11) as

$$p(x) = \sum_{k=1}^{K} c_k x^{\gamma_k} \tag{C.1}$$

where $c = [c_1, c_2, \cdots, c_K]$ is the coefficient vector, $x \in \mathbb{R}^n$, and

$$\gamma = \begin{bmatrix} \gamma_1(1) & \gamma_2(1) & \cdots & \gamma_K(1) \\ \gamma_1(2) & \gamma_2(2) & \cdots & \gamma_K(2) \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_1(n) & \gamma_2(n) & \cdots & \gamma_K(n) \end{bmatrix} \tag{C.2}$$

$$x^{\gamma_k} = x_1^{\gamma_k(1)} x_2^{\gamma_k(2)} \cdots x_n^{\gamma_k(n)}. \tag{C.3}$$

The process of dimension elevation consists of adding one row of zeros in the exponent matrix γ and treat $x \in \mathbb{R}^{n+1}$. Since all the monomials have the term x_{n+1}^0 this new exponent matrix define the same polynomial, except for it being defined in the artificial dimension too. This makes the description in the Bernstein basis have more coefficients.

To see the equivalence, first consider Figure C.1 where the two processes are outlined for the one dimensional case. The simplex σ^1 is either sub-divided into σ^2 and σ^3, or dimension elevated to σ^4. Note that the vertex numbering is different from the rest of the thesis to ease the notation. The vertices σ_i are with either one or two coordinates depending on them being

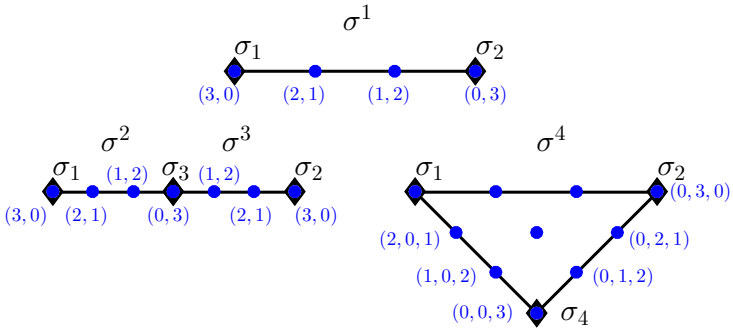


Figure C.1: Simplex σ^1 under sub-division into σ^2 and σ^3 or dimension elevation into σ^4. Note that the vertex numbering is different from the rest of the thesis.

defined in one or two dimensions. Most of the grid points are numbered but the Δ is omitted. The shown degree is $D = 3$, but this is arbitrary and only serves to exemplify the process.

To see the equality, next consider the Bernstein coefficient $b_{(2,1)}(\sigma^2)$, which from Equation (B.17) is given as

$$b_{(2,1)}(\sigma^2) = \sum_{k=1}^{K} c_k \sum_{\substack{|\beta|=\gamma_k(1) \\ (2,1)-\beta\geq 0}} \sigma_1(1)^{\beta_1}\sigma_3(1)^{\beta_2} \frac{\binom{\gamma_k(1)}{\beta}\binom{D-\gamma_k(1)}{(2,1)-\beta}}{\binom{D}{(2,1)}}, \tag{C.4}$$

with $\beta \in \mathbb{N}_0^{n+1}$. Note that on σ^2, σ_1 and σ_3 only have one coordinate.

Consider now the Bernstein coefficient $b_{(2,0,1)}(\sigma^4)$, which from Equation (B.17) is given as

$$\begin{aligned} b_{(2,0,1)}(\sigma^4) = \sum_{k=1}^{K} c_k \sum_{\substack{|\kappa|=\gamma_k(1)+\gamma_k(2) \\ (2,0,1)-\kappa\geq 0}} \sum_{\substack{|\kappa-\beta|=\gamma_k(2) \\ \kappa-\beta\geq 0}} \sigma_1(1)^{\beta_1}\sigma_2(1)^{\beta_2}\sigma_4(1)^{\beta_3} \\ \sigma_1(2)^{\kappa_1-\beta_1}\sigma_2(2)^{\kappa_2-\beta_2}\sigma_4(2)^{\kappa_3-\beta_3} \frac{\binom{\gamma_k(1)}{\beta}\binom{\gamma_k(2)}{\kappa-\beta}\binom{D-(\gamma_k(1)+\gamma_k(2))}{(2,0,1)-\kappa}}{\binom{D}{(2,0,1)}}. \end{aligned} \tag{C.5}$$

First note, by construction, that $\gamma_k(2) = 0\ \forall k$. This is the result of adding the row of zeros for the existing monomial combinations in the exponent matrix. Simplifying yields

$$b_{(2,0,1)}(\sigma^4) = \sum_{k=1}^{K} c_k \sum_{\substack{|\kappa|=\gamma_k(1) \\ (2,0,1)-\kappa\geq 0}} \sum_{\substack{|\kappa-\beta|=0 \\ \kappa-\beta\geq 0}} \sigma_1(1)^{\beta_1}\sigma_2(1)^{\beta_2}\sigma_4(1)^{\beta_3} \frac{\binom{\gamma_k(1)}{\beta}\binom{D-\gamma_k(1)}{(2,0,1)-\kappa}}{\binom{D}{(2,0,1)}}. \tag{C.6}$$

Next, note that $\sum_{|\kappa-\beta|=0}$ is a sum over one element, with $\kappa = \beta$. Substituting gives

$$b_{(2,0,1)}(\sigma^4) = \sum_{k=1}^{K} c_k \sum_{\substack{|\beta|=\gamma_k(1) \\ (2,0,1)-\beta\geq 0}} \sigma_1(1)^{\beta_1}\sigma_2(1)^{\beta_2}\sigma_4(1)^{\beta_3} \frac{\binom{\gamma_k(1)}{\beta}\binom{D-\gamma_k(1)}{(2,0,1)-\beta}}{\binom{D}{(2,0,1)}}. \quad \text{(C.7)}$$

The requirement $(2,0,1)-\beta \geq 0$ forces $\beta_2 = 0$ for the picked grid point. This gives the final simplification

$$b_{(2,0,1)}(\sigma^4) = \sum_{k=1}^{K} c_k \sum_{\substack{|\beta|=\gamma_k(1) \\ (2,0,1)-\beta\geq 0}} \sigma_1(1)^{\beta_1}\sigma_4(1)^{\beta_3} \frac{\binom{\gamma_k(1)}{\beta}\binom{D-\gamma_k(1)}{(2,0,1)-\beta}}{\binom{D}{(2,0,1)}}. \quad \text{(C.8)}$$

Now, the only difference between Equation (C.4) and Equation (C.8) is $\sigma_3(1)$ and $\sigma_4(1)$. Thus sub-division is equivalent to dimension elevation if the new vertex from sub-division and the first coordinate of the extra vertex from dimension elevation coincide.

Note that on σ^4, for grid points on the face defined by vertices σ_1 and σ_4, the second element in the grid point will always be equal to zero. For grid points on the face defined by vertices σ_2 and σ_4, the first element in the grid point will always be equal to zero. This gives

$$\begin{aligned} &\left[b_{(3,0)}(\sigma^2), b_{(2,1)}(\sigma^2), b_{(1,2)}(\sigma^2), b_{(0,3)}(\sigma^2)\right] = \\ &\left[b_{(3,0,0)}(\sigma^4), b_{(2,0,1)}(\sigma^4), b_{(1,0,2)}(\sigma^4), b_{(0,0,3)}(\sigma^4)\right] \end{aligned} \quad \text{(C.9)}$$

and

$$\begin{aligned} &\left[b_{(3,0)}(\sigma^3), b_{(2,1)}(\sigma^3), b_{(1,2)}(\sigma^3), b_{(0,3)}(\sigma^3)\right] = \\ &\left[b_{(0,3,0)}(\sigma^4), b_{(0,2,1)}(\sigma^4), b_{(0,1,2)}(\sigma^4), b_{(0,0,3)}(\sigma^4)\right]. \end{aligned} \quad \text{(C.10)}$$

If one wishes to sub-divide more than once, an equal number of artificial dimension can be added to obtain a result similar to the above. Also, the arguments presented above extends naturally to higher dimension.

Appendix C. Positivity Certification By Dimension Elevation

Appendix D

Investigation of Counterexample

This Appendix is concerned with the gap between a polynomial being positive definite and being Bernstein basis certifiable, as introduced in Section 14, and in particular an explicit counterexample given in [45].

Counterexample

The polynomial

$$\begin{aligned} p =& 21x_1^4 + 24x_1^3x_2 - 36x_1^3 + 18x_1^2x_2^2 - \\ & 24x_1^2x_2 + 18x_1^2 + 12x_1x_2^3 - 12x_1x_2^2 + 30x_2^4 \end{aligned} \tag{D.1}$$

can be written as a sum-of-squares (SOS) like

$$p = \begin{bmatrix} x_1 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}^T \begin{bmatrix} 18 & -18 & -12 & -6 \\ -18 & 21 & 12 & 0 \\ -12 & 12 & 18 & 6 \\ -6 & 0 & 6 & 30 \end{bmatrix} \begin{bmatrix} x_1 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{D.2}$$

The matrix has positive eigenvalues making p positive definite. This is also seen by the Cholesky factorisation leaving

$$p = \sum_{i=1}^{4} g_i^2 \tag{D.3}$$

with

$$g_1 = -4.2426x_1^2 - 2.8284x_1x_2 + 4.2426x_1 - 1.4142x_2^2 \tag{D.4}$$

$$g_2 = 1.7321x_1^2 - 3.4641x_2^2 \tag{D.5}$$

$$g_3 = 0.63246x_2^2 + 3.1623x_1x_2 \tag{D.6}$$

$$g_4 = 3.9497x_2^2. \tag{D.7}$$

Here it is easily seen from g_2 and g_4 than p has a unique minimum of 0, at the origin.

When described on the standard simplex, see Equation (25), the coefficient vector is

$$b(p, 4, \sigma) = [0, 0, 0, 3, 0, 0, 0, 1, -1, 0, 3, 0, 0, 0, 30]. \tag{D.8}$$

In [45] it is proven that sub-dividing cannot change the sign of the negative coefficient for the simplex with the origin as one of its vertices. The same is true if the degree of the representation is raised, the coefficient remains negative. This shows that p is an example of the gab between positive definite and Bernstein basis certifiable.

Following the discussion on existence and solvability in Section 20.1, the rest of this Appendix is concerned with the following question: Can the polynomial p serve as a Lyapunov function for a stable dynamical system described by a polynomial vector field? And if so, can that system only be shown stable using polynomials which do not admit to Bernstein basis certifying collections?

The conclusion is, that the polynomial can serve as a Lyapunov function for a synthesised stable system, but that that system also admits to other Lyapunov functions which are Bernstein basis certifiable.

Stable Polynomial Vector Field

In this Section the vector field is synthesised. Seeing as the Lyapunov function is given only the following equation has to be true for the vector field.

Let $\dot{x} = f(x)$ be the stable vector field ($f(0) = 0$ is the equilibrium), and let the polynomial p in Equation (D.1) be a Lyapunov function for it. Then

$$\frac{\partial V(x)}{\partial x} f(x) < 0 \qquad \forall x \neq 0 \tag{D.9}$$

needs to be fulfilled. This inequality is relaxed as an SOS-criteria. An SOS-program is set up and solved using SOSTOOLS [40], and it produces the

following system

$$\begin{aligned} f_1 =& -18.13x_1^3 - 1.193x_1^2x_2 - 14.77x_1x_2^2 - 0.1836x_2^3 \\ & - 23.67x_1^2 - 12.51x_1x_2 - 8.56x_2^2 - 18.9x_1 - 0.0003419x_2, \\ f_2 =& 15.46x_1^3 + 0.01834x_1^2x_2 + 3.458x_1x_2^2 - 8.342x_2^3 + \\ & 13.94x_1^2 + 0.9568x_1x_2 + 0.5464x_2^2 + 1.966x_1 - 18.23x_2. \end{aligned} \tag{D.10}$$

This proves p's ability to serve as a Lyapunov function.

Synthesising using the BBAlgorithm

In this Section the vector field (D.10) is shown stable using the BBAlgoritm.

Using a polynomial Lyapunov function of degree $d_V = 4$ the BBAlgorithm retunes

$$\begin{aligned} V =& 2.2x_1^4 + 2.4x_1^3x_2 + 0.085x_1^3 + 8.1x_1^2x_2^2 + 0.13x_1^2x_2 + 0.12x_1^2 + \\ & 2.2x_1x_2^3 + 0.18x_1x_2^2 + 0.079x_1x_2 + 2.1x_2^4 - 0.045x_2^3 + 0.12x_2^2. \end{aligned} \tag{D.11}$$

This proves that the system allows for Lyapunov functions which admits to Bernstein basis certifying collections, even though it also (by design) admits to at least one Lyapunov function which does not admits a Bernstein basis certifying collection.

Synthesising using an SOS-Program

In this Section the vector field (D.10) is shown stable using an SOS realisation. This is trivial since this is how the vector field was designed in the first place, but it is interesting to see if an SOS-program finds the same Lyapunov function, e.g. (D.1), or if it finds another polynomial.

For this synthesis, the following two criteria needs to be fulfilled:

$$V(x) > 0 \qquad \forall x \neq 0 \wedge V(0) = 0, \tag{D.12}$$

$$\frac{\partial V(x)}{\partial x} f(x) < 0 \qquad \forall x \neq 0 \wedge \dot{V}(0) = 0, \tag{D.13}$$

where V is the unknown, and $f(x)$ is from (D.10). The inequalities are relaxed as SOS-criteria. An SOS-program is set up and solved, and produces the following Lyapunov function:

$$\begin{aligned} V(x) =& 1.87x_1^4 + 0.1955x_1^3x_2 - 0.5042x_1^3 + 1.316x_1^2x_2^2 - \\ & 0.009137x_1^2x_2 + 2.397x_1^2 - 0.001332x_1x_2^3 - 0.1279x_1x_2^2 + \\ & 0.05443x_1x_2 + 1.636x_2^4 - 0.0439x_2^3 + 2.403x_2^2. \end{aligned} \tag{D.14}$$

This proves that the synthesised vector field allows for a Lyapunov function found using SOS realisations which is different than the SOS polynomial used to create it. In addition, the SOS synthesised Lyapunov function in Equation (D.14) can be certified positive definite using the function *getPositivityCertificate* from Chapter Positivity Certification (and the Lie derivative certified negative definite). This shows that it is not an inherent feature of the synthesised vector field that it needs SOS-Lyapunov functions without Bernstein basis certifying collections. This is evidence towards that polynomials like (D.1) are rare in the sense of their role in Lyapunov function analysis.

Appendix E

Polynomial Lyapunov Functions for Rationally Stable Systems

This Appendix presents a proof that rationally stable dynamical systems admits polynomial Lyapunov functions on bounded regions. It was first presented in [28]. The proof is heavily inspired by the proof from [38] for exponential stability, where the weight $1/||x||_2^2$ is of eminent importance. This result is extended to include the weight $1/||x||_2^r$ with $r \in \mathbb{R}_{>0}$. Before stating the proof some definitions are needed.

Definitions

Denote the set of continuous maps defined on $\Omega \subset \mathbb{R}^n$ by $C(\Omega)$ with norm

$$||f||_\infty = \sup_{x \in \Omega} ||f(x)||_\infty. \tag{E.1}$$

Note for a function, (E.1) boils down to $\sup_{x\in\Omega} |f(x)|$.

For operators $h_i : X \to X$, let $\prod_i h_i : X \to X$ denote the sequential composition of the h_i, i.e.,

$$\prod_i h_i = h_1 \circ h_2 \circ \cdots \circ h_{n-1} \circ h_n. \tag{E.2}$$

Following [38], for $\Omega \in \mathbb{R}^n$, define the following sets of differentiable functions:

$$C_1^i(\Omega) = \left\{ f : D^\alpha f \in C(\Omega) \ \forall \ \alpha \in \mathbb{N}^n \text{ s. t. } \sum_{j=1}^n \alpha_j \leq i \right\},$$

$$C^i_\infty(\Omega) = \left\{ f : D^\alpha f \in C(\Omega) \ \forall \ \alpha \in \mathbb{N}^n \text{ s. t. } \max_j \alpha_j \leq i \right\}.$$

The following definition is from [38].

Definition 39 *An element of X is defined to be a set of 2^n continuous functions, indexed as $f_\alpha \in C(\Omega)$ for $\alpha \in \{0,1\}^n$, and denoted $\{f_\alpha\}_{\alpha\in\{0,1\}^n} \in X$. Define the linear map $K : X \to C^1_\infty(\Omega)$ as*

$$K(\{f_\alpha\}_{\alpha\in\{0,1\}^n}) = \sum_{\alpha\in\{0,1\}^n} G_\alpha f_\alpha \tag{E.3}$$

where $G_\alpha : C(\Omega) \to C^1_\infty(\Omega)$ is given by

$$G_\alpha h = \left(\prod_{i=1}^n g_{i,\alpha_i} \right) h \tag{E.4}$$

and where the operators $g_{i,j}$ are given by

$$(g_{i,j}h)(x_1,\ldots,x_n) = \begin{cases} h(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) & ,j=0 \\ \int_0^{x_i} h(x_1,\ldots,x_{i-1},s,x_{i+1},\ldots,x_n)ds & ,j=1. \end{cases} \tag{E.5}$$

Lemmas

Definition 39 has the following consequence, given as Lemma 3 in [38].

Lemma 40 *([38]) For $v \in C^1_\infty(\Omega)$, $K(\{D^\alpha v\}_{\alpha\in\{0,1\}^n}) = v$.*

Lemma 41 *For $r \in \mathbb{R}_{>0}$, let $\hat{r} = \lceil r \rceil$ be the smallest integer bigger than r. Let $v \in C^{\hat{r}}_1(B^\infty)$. Then for any $\epsilon > 0$ there exists a polynomial p and an open neighbourhood U of 0 such that*

$$\left|\left| \frac{p(x) - v(x)}{||x||_2^r} \right|\right|_\infty \leq \epsilon, \qquad \forall x \in U. \tag{E.6}$$

Proof
Let the polynomial m be the $\hat{r}^{\text{th}}$ order Taylor series expansion of v about $x = 0$

$$\begin{aligned} m(x) = v(0) + \sum_{i=1}^n x_i \frac{\partial v}{\partial x_i}(0) + \frac{1}{2} \sum_{i_1,i_2=1}^n x_{i_1} x_{i_2} \frac{\partial^2 v}{\partial x_{i_1} \partial x_{i_2}}(0) \\ + \cdots + \frac{1}{\hat{r}!} \sum_{i_1,\ldots,i_{\hat{r}}=1}^n x_{i_1} \cdots x_{i_{\hat{r}}} \frac{\partial^{\hat{r}} v}{\partial x_{i_1} \cdots \partial x_{i_{\hat{r}}}}(0). \end{aligned} \tag{E.7}$$

Then by continuity, m approximates v close to the origin and

$$(v-m)(0) = \frac{\partial (v-m)}{\partial x_i}(0) = \cdots = \frac{\partial^{\hat{r}} (v-m)}{\partial x_{i_1} \cdots \partial x_{i_{\hat{r}}}}(0) = 0. \tag{E.8}$$

Let $w(x) = ||x||_2^r$ and define

$$h(x) = \begin{cases} 0 & , x = 0 \\ \frac{v(x) - m(x)}{w(x)} & , \text{otherwise.} \end{cases} \tag{E.9}$$

Since

$$v(x) = m(x) + R_{v,\hat{r}}(x), \tag{E.10}$$

h can be written as

$$h(x) = \begin{cases} 0 & , x = 0 \\ \frac{R_{v,\hat{r}}(x)}{w(x)} & , \text{otherwise.} \end{cases} \tag{E.11}$$

Since the degree of $R_{v,\hat{r}}(x)$ is strictly greater than the degree of $w(x)$

$$\lim_{x \to 0} \left(\frac{R_{v,\hat{r}}(x)}{w(x)} \right) = 0, \tag{E.12}$$

and h is continuous at 0. By Weierstrass approximation theorem, for every $\epsilon/2 > 0$ there exists a polynomial q such

$$||q - h||_\infty \leq \epsilon/2. \tag{E.13}$$

Let $p(x) = m(x) + q(x)[||x||_2^r - R_{w,\hat{r}}(x)]$ where $R_{w,\hat{r}}(x)$ is the remainder after an $\hat{r}^{\text{th}}$ order Taylor series expansion of $||x||_2^r$ about $x = 0$. Finally

$$\left|\left| \frac{p(x) - v(x)}{w(x)} \right|\right|_\infty = \left|\left| \frac{m(x) + q(x)w(x) - q(x)R_{w,\hat{r}}(x) - v(x)}{w(x)} \right|\right|_\infty \tag{E.14}$$

$$= \left|\left| \frac{m(x) - v(x) - q(x)R_{w,\hat{r}}(x)}{w(x)} + h(x) + (q(x) - h(x)) \right|\right|_\infty \tag{E.15}$$

$$= \left|\left| \frac{-q(x)R_{w,\hat{r}}(x)}{w(x)} + \underbrace{\frac{m(x) - v(x)}{w(x)} + h(x)}_{=0} + (q(x) - h(x)) \right|\right|_\infty \tag{E.16}$$

$$\leq \left|\left| \frac{-q(x)R_{w,\hat{r}}(x)}{w(x)} \right|\right|_\infty + ||(q(x) - h(x))||_\infty \tag{E.17}$$

$$\leq \left|\left| \frac{-q(x)R_{w,\hat{r}}(x)}{w(x)} \right|\right|_\infty + \epsilon/2 \tag{E.18}$$

where the first inequality is due to the triangle inequality on norms and the second inequality is from Equation (E.13). Now, since $\hat{r} \geq r$

$$\lim_{x \to 0} \frac{-q(x)R_{w,\hat{r}}(x)}{||x||_2^r} = 0. \tag{E.19}$$

Thus there is a sufficiently small region around 0 such that

$$\left|\left|\frac{p(x)-v(x)}{||x||_2^r}\right|\right|_\infty \leq \epsilon/2+\epsilon/2=\epsilon. \tag{E.20}$$

■

Lemma 42 *Let* $p=\{p_\alpha\}_{\alpha\in\{0,1\}^n}$ *and* $q=\{q_\alpha\}_{\alpha\in\{0,1\}^n}$ *with* $p_\alpha, q_\alpha \in C(B^\infty)$. *Then*

$$\max_{\beta\in\{0,1\}^n}\left|\left|\frac{D^\beta Kp(x)-D^\beta Kq(x)}{||x||_2^r}\right|\right|_\infty \leq 2^n \max_{\alpha\in\{0,1\}^n}\left|\left|\frac{p_\alpha(x)-q_\alpha(x)}{||x||_2^r}\right|\right|_\infty. \tag{E.21}$$

Proof
By the definition of $g_{j,k}$

$$\frac{\partial}{\partial x_i}g_{j,k}f=\begin{cases} g_{j,k}\frac{\partial}{\partial x_i}f & ,i\neq j \\ f & ,i=j,k=1 \\ 0 & ,i=j,k=0 \end{cases} \tag{E.22}$$

which implies

$$\frac{\partial^\beta}{\partial x^\beta}G_\alpha f=\begin{cases} 0 & ,\alpha_i<\beta_i \text{ for some } i \\ \left(\prod_{\substack{i=1\\ \beta_i\neq 1}}^n g_{i,\alpha_i}\right)f & ,\text{otherwise.} \end{cases} \tag{E.23}$$

The first step in the proof is to obtain bounds on the function

$$\frac{1}{||x||_2^r}(g_{i,j}f)(x). \tag{E.24}$$

If $j=0$ and $x\in B^\infty$, then for $n=1$,

$$\left|\left|\frac{1}{||x||_2^r}(g_{1,0}f)(x)\right|\right| \leq \left|\left|\frac{f(x)}{||x||_2^r}\right|\right|_\infty \tag{E.25}$$

follows directly from Definition 39. For $n>1$

$$\left|\frac{1}{||x||_2^r}(g_{i,0}f)(x)\right| = \left|\frac{f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)}{||x||_2^r}\right| \tag{E.26}$$

$$\leq \left|\frac{f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)}{\left(\sum_{\substack{k=1\\ k\neq i}}^n x_k^2\right)^{r/2}}\right| \tag{E.27}$$

$$\leq \left|\left|\frac{f(s)}{||s||_2^r}\right|\right|_\infty, \tag{E.28}$$

where the first inequality is due to $x_i^2 \geq 0$. If $j = 1$ and $x \in B^\infty$, then for $n \geq 1$

$$\left|\frac{1}{||x||_2^r}(g_{i,1}f)(x)\right| = \left|\int_0^{x_i} \frac{f(x_1,\ldots,x_{i-1},t,x_{i+1},\ldots,x_n)}{||x||_2^r}dt\right| \tag{E.29}$$

$$\leq \sup_{\nu\in[-|x_i|,|x_i|]} \frac{|f(x_1,\ldots,x_{i-1},\nu,x_{i+1},\ldots,x_n)|}{\left(\sum_{k=1}^n x_k^2\right)^{r/2}} \tag{E.30}$$

$$\leq \sup_{\nu\in[-|x_i|,|x_i|]} \frac{|f(x_1,\ldots,x_{i-1},\nu,x_{i+1},\ldots,x_n)|}{\left(\nu + \sum_{\substack{k=1\\k\neq i}}^n x_k^2\right)^{r/2}} \tag{E.31}$$

$$\leq \left|\left|\frac{f(s)}{||s||_2^r}\right|\right|_\infty. \tag{E.32}$$

The first inequality it due to the mean value theorem and that $|x_i| \leq 1$, and the second follows from $x_i^2 \geq \nu^2$. This gives

$$\left|\left|\frac{1}{||x||_2^r}(g_{i,1}f)(x)\right|\right|_\infty \leq \left|\left|\frac{1}{||x||_2^r}f(x)\right|\right|_\infty. \tag{E.33}$$

Thus, for $j \in \{0,1\}$ and $i = 1,\ldots,n$

$$\left|\left|\frac{1}{||x||_2^r}(g_{i,j}f)(x)\right|\right|_\infty \leq \left|\left|\frac{1}{||x||_2^r}f(x)\right|\right|_\infty. \tag{E.34}$$

This bound can be applied inductively since the terms G_α are compositions of $g_{i,j}$. For any $\beta \in \{0,1\}^n$ this gives

$$\left|\left|\frac{1}{||x||_2^r}\left(\frac{\partial^\beta}{\partial x^\beta}G_\alpha f\right)(x)\right|\right|_\infty = \left|\left|\frac{1}{||x||_2^r}\left(\left(\prod_{\substack{i=1\\\beta_i\neq 1}}^n g_{i,\alpha_i}\right)f\right)(x)\right|\right|_\infty \tag{E.35}$$

$$\leq \left|\left|\frac{1}{||x||_2^r}\left(\left(\prod_{\substack{i=2\\\beta_i\neq 1}}^n g_{i,\alpha_i}\right)f\right)(x)\right|\right|_\infty \tag{E.36}$$

$$\cdots \leq \left|\left|\frac{f(x)}{||x||_2^r}\right|\right|_\infty. \tag{E.37}$$

Given this bound on G_α, for any $\beta \in \{0,1\}^n$, the triangle inequality yields

$$\left|\left|\frac{D^\beta Kp(x) - D^\beta Kq(x)}{||x||_2^r}\right|\right|_\infty = \left|\left|\frac{1}{||x||_2^r}\frac{\partial^\beta}{\partial x^\beta}K(p-q)(x)\right|\right|_\infty \tag{E.38}$$

$$\leq \sum_{\alpha\in\{0,1\}^n}\left|\left|\frac{1}{||x||_2^r}\left(\frac{\partial^\beta}{\partial x^\beta}G_\alpha(p_\alpha - q_\alpha)\right)(x)\right|\right|_\infty \tag{E.39}$$

$$\leq \sum_{\alpha\in\{0,1\}^n}\left|\left|\frac{p_\alpha(x) - q_\alpha(x)}{||x||_2^r}\right|\right|_\infty \tag{E.40}$$

$$\leq 2^n \max_{\alpha\in\{0,1\}^n}\left|\left|\frac{p_\alpha(x) - q_\alpha(x)}{||x||_2^r}\right|\right|_\infty, \tag{E.41}$$

where the second inequality is from the bound in (E.37) and the third inequality is from the fact that p_α and q_α both have 2^n elements.

■

Lemma 43 *Let v be a function with partial derivatives*

$$D^\alpha v \in C_1^2(B^\infty) \tag{E.42}$$

for all $\alpha \in \{0,1\}^n$. Then for $r \in \mathbb{R}_{>0}$ and for any $\epsilon > 0$, there exists a polynomial p, such that for $x \in B^\infty$

$$\max_{\alpha\in\{0,1\}^n}\left|\left|\frac{D^\alpha p(x) - D^\alpha v(x)}{||x||_2^r}\right|\right|_\infty \leq \epsilon. \tag{E.43}$$

Proof
From Lemma 41 there exists polynomial functions q_α such that

$$\max_{\alpha\in\{0,1\}^n}\left|\left|\frac{q_\alpha(x) - D^\alpha v(x)}{||x||_2^r}\right|\right|_\infty \leq \frac{\epsilon}{2^n}. \tag{E.44}$$

Let $q = \{q_\alpha\}_{\alpha\in\{0,1\}^n}$ and $p = Kq$. Since the q_α are polynomial, p is polynomial. Let $h = \{D^\alpha v\}_{\alpha\in\{0,1\}^n}$. Then by Lemma 40, $v = Kh$. Therefore by Lemma 42

$$\max_{\alpha\in\{0,1\}^n}\left|\left|\frac{D^\alpha p(x) - D^\alpha v(x)}{||x||_2^r}\right|\right|_\infty = \max_{\alpha\in\{0,1\}^n}\left|\left|\frac{D^\alpha Kr(x) - D^\alpha Kh(x)}{||x||_2^r}\right|\right|_\infty \tag{E.45}$$

$$\leq 2^n \max_{\alpha\in\{0,1\}^n}\left|\left|\frac{q_\alpha(x) - D^\alpha v(x)}{||x||_2^r}\right|\right|_\infty \tag{E.46}$$

$$\leq \epsilon, \tag{E.47}$$

thus proving the Lemma.

■

Theorem

Combining the above the result yields the counterpart to Theorem 9 in [38].

Theorem 44 *(Rationally Stable Vector Fields have Polynomial Lyapunov Functions on Bounded Regions.) Let $\Omega \subseteq B^\infty$ and let f be a given general non-linear vector field uniformly bounded on B^∞. Suppose there exists a function $V : B^\infty \to \mathbb{R}$ with $D^\alpha V \in C_1^2(B^\infty)$ for all $\alpha \in \{0,1\}^n$ and such that*

$$\alpha_0 ||x||_2^{r_1} \leq V(x) \leq \beta_0 ||x||_2^{r_2} \tag{E.48}$$

$$\nabla V(x)^T f(x) \leq -\gamma_0 ||x||_2^{r_3}, \tag{E.49}$$

where $\alpha_0, \beta_0, \gamma_0, r_1, r_2, r_3 \in \mathbb{R}_{>0}$ are constants with $r_3 > r_2$. Then there exists a polynomial p such that

$$\alpha ||x||_2^{r_1} \leq p(x) \leq \beta ||x||_2^{r_2} \tag{E.50}$$

$$\nabla p(x)^T f(x) \leq -\gamma ||x||_2^{r_3}, \tag{E.51}$$

with $\alpha_0 > \alpha > 0$, $0 < \beta_0 < \beta$ and $\gamma_0 > \gamma > 0$.

Proof
Let $b = ||f||_\infty = \sup_{x \in B^\infty} ||f(x)||_\infty$ and choose $0 < \epsilon < \min\{\alpha_0 - \alpha, \beta - \beta_0, (\gamma_0 - \gamma)/(nb)\}$. By Lemma 43 there exists a polynomial p such that

$$\max_{r \in \{r_1, r_2, r_3\}} \left| \frac{p(x) - V(x)}{||x||_2^r} \right| \leq \epsilon, \qquad \max_{r \in \{r_1, r_2, r_3\}} \left| \frac{\frac{\partial p}{\partial x_i}(x) - \frac{\partial V}{\partial x_i}(x)}{||x||_2^r} \right| \leq \epsilon, \tag{E.52}$$

for all $i = 1, \ldots, n$. This gives the following for the conditions on p:

$$p(x) = V(x) + \frac{p(x) - V(x)}{||x||_2^{r_1}} ||x||_2^{r_1} \tag{E.53}$$

$$\geq \alpha_0 ||x||_2^{r_1} - \epsilon ||x||_2^{r_1} \tag{E.54}$$

$$\geq \alpha ||x||_2^{r_1}, \tag{E.55}$$

and

$$p(x) = V(x) + \frac{p(x) - V(x)}{||x||_2^{r_2}} ||x||_2^{r_2} \tag{E.56}$$

$$\leq \beta_0 ||x||_2^{r_2} + \epsilon ||x||_2^{r_2} \tag{E.57}$$

$$\leq \beta ||x||_2^{r_2}. \tag{E.58}$$

For the derivative it becomes:

$$\begin{aligned}
\nabla p(x)^T f(x) &= \nabla (p(x) - V(x))^T f(x) + \nabla V(x)^T f(x) && \text{(E.59)}\\
&= \frac{\nabla (p(x) - V(x))^T f(x)}{||x||_2^{r_3}} ||x||_2^{r_3} + \nabla V(x)^T f(x) && \text{(E.60)}\\
&\leq \sum_{i=1}^{n} \frac{\frac{\partial p}{\partial x_i}(x) - \frac{\partial V}{\partial x_i}(x)}{||x||_2^{r_3}} f_i(x) ||x||_2^{r_3} - \gamma_0 ||x||_2^{r_3} && \text{(E.61)}\\
&\leq \epsilon n b ||x||_2^{r_3} - \gamma_0 ||x||_2^{r_3} && \text{(E.62)}\\
&\leq -\gamma ||x||_2^{r_3}. && \text{(E.63)}
\end{aligned}$$

■

Appendix F

Software Tutorial

This Appendix covers the usage of some of the functions introduced in the thesis. The functions covered are the ones most usable as stand alone functions. It was chosen to group similar functions together despite them being introduced in different chapters. Each section is accompanied by a tutorial function which can be executed prior to or in addition to reading the sections.

Collections

The function *tutorialCollections* covers the introduction presented in this Section. To get the initial collection of simplices covering a hypercube, to subdivide it, and to plot the collections, the functions *getInitialPartition*, *getBinarySplitting*, and *plotCollection* are used.

As an example, define an interval as

```
I = [-1 1;-1 1;-1 1];
```

and obtain the initial collection like

```
[vex,simplex] = getInitialPartition(I);.
```

To visualise the collection call

```
plotCollection(vex,simplex).
```

To sub-divide all 12 simplices in the collection call

```
[vex,simplex] = getBinarySplitting(vex,simplex);
```

or call e.g.

```
[vex,simplex] = getBinarySplitting(vex,simplex,[1 6 7]);
```

to sub-divide the first, the sixth and the seventh simplex. Use *plotCollection* again to plot the result. Try different hypercubes.

Basis Transformation

The function *tutorialBasisTransformation* covers the introduction presented in this Section. The Bernstein basis polynomials of a given degree on a given simplex are obtained using *getBernsteinBasisPolynomials*. Define the two dimensional standard simplex as

```
vex = [0 1 0;0 0 1];
```

and call

```
BBP = getBernsteinBasisPolynomials(2,vex);
```

to get the basis polynomials of degree 2 on the simplex. Call

```
BBP = getBernsteinBasisPolynomials(10,vex);
```

to get the basis polynomials of degree 10 on the simplex, and so on. Try different simplices.

To transform between monomial and Bernstein bases use *getMon2BernTrans*. Define a polynomial in the monomial basis as

```
c = [3 4 9 1 -5];
gamma = [3 1 1 0 0;0 2 1 3 1];
```

and use *getInitialPartition* to get a collection of simplices on the two dimensional unit cube as described above. To get the transformation matrix call

```
[M2B,alpha,d,simplexCtrlPointsVF] = ...
getMon2BernTrans(2,gamma,vex,simplex);
```

and to get the Bernstein coefficients call

```
C = (M2B*c.').';.
```

To get the coefficients on the first simplex call

```
b1 = C(simplexCtrlPointsVF(1,:));
```

and to check the result recover the monomial description calling

```
BBP = getBernsteinBasisPolynomials(d,vex(:,simplex(1,:)));
p = b1*BBP;.
```

Try doing this for all simplices.

To transform between descriptions on different simplices use *getCtrlPoints-BernTrans*. Still working on the collection of simplices covering the two dimensional unit cube from above, define a polynomial of degree 3 in the Bernstein basis on the fourth simplex as

```
b4 = [1 2 0 -4 3 -8 4 0 1 2];.
```

Get the transformation matrix by calling

```
[B2C,alpha,d,simplexCtrlPointsVF] = ...
getCtrlPointBernTrans(2,3,vex,simplex);
```

and get the coefficient vector from the call

```
C = (B2C*b4.').';.
```

To visualise the polynomial on the collection call

```
for i = 1:4
getLyapunovFunctionPlot(C(simplexCtrlPointsVF(i,:)), ...
alpha,d,vex(:,simplex(i,:)))
end.
```

Certificates

The function *tutorialCertificates* covers the introduction presented in this Section. To certify the positivity of a polynomial use *getPositivityCertificate*. Create a collection of simplices, e.g.

```
I = [-1 1;-2 3;]; [vex,simplex] = getInitialPartition(I);,
```

and define a polynomial in the monomial basis as

```
c = [2401 -1078 -8993 2046 8649 3822 -1642 -7078 1488 -5045 ...
     850 12526 -5226 1072 4492];
gamma = [4 3 2 1 0 3 2 1 0 2 1 0 1 0 0;
         0 1 2 3 4 0 1 2 3 0 1 2 0 1 0];.
```

The call

```
[C,alpha,d,n,vex,simplex] = ...
getPositivityCertificate(c,gamma,vex,simplex);
```

certifies the positivity of the polynomial. The function sub-divides the collection into a total of 59 simplices before the certificate is obtained. The certificate is the fact that C only has positive entries. Plotting the collection using *plotCollection* reveals the sub-division to be highly concentrated.

To certify the stability of a polynomial vector field use *analyseStability*. It requires a working installation of the MOSEK software, visit mosek.com for more information. Define a polynomial vector field like

```
c =     [-1 2 -1 4 -8  4 -1 4  0 -4  0 10  0;
          0 0  0 0 -9 10  0 2 -8 -4 -1  4 -4];
gamma = [3 3  3 2  2  2  1 1  1  1  0  0  0;
         2 1  0 2  1  0  4 3  2  0  3  2  1];
```

and a hypercube

```
I = [-1 1;-1 1];.
```

The call

```
[result,vex,simplex,CV,alphaV,dV,simplexCtrlPointsV,CL, ...
                                                alphaLie,dLie] = ...
analyseStability(c,gamma,I);
```

certifies the stability of the vector field and returns the conclusion, the synthesised Lyapunov function and the collection on which it is described. The function *analyseStability* automatically plots the Lyapunov function and the Lie derivative, if the vector field is two dimensional. Try expanding the hypercube to

```
I = [-2 2;-2 2];
```

to require sub-division. Also try changing the degree and type of Lyapunov function and to certify the instability of the unstable system with the call

```
[result,vex,simplex,CV,alphaV,dV,simplexCtrlPointsV,CL, ...
                                                alphaLie,dLie] = ...
analyseStability(-c,gamma,I,2,'POL','ALL','INS');.
```

The *.m*-script *sampleSystems* contains a collection of polynomial vector fields which are stable. They can be used to further familiarise one self with the *analyseStability* function. Notice in particular benchmark 13 which initially took 80 hours to solve. After an update to MATLAB 2016a in which the execution engine was altered, the system is now certified stable in mere minutes(!).

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-087-6

AALBORG UNIVERSITY PRESS